

CHAPTER 7

FOSI-Generated Material

A FOSI can generate content that is inserted into the output structure. Details are furnished in the following sections:

- **Generated text** on page 120
- **Generated tables** on page 129
- **Generated graphics** on page 132
- **Generated files** on page 134
- **Generated HTML** on page 136
- **Generated sorted/merged/grouped material (indexing)** on page 139

Generated text

Usually, **FOSI-generated text** (**gentext**) is output from a `usetext` category. `Sectext` also outputs generated text. However, assigning a value for `sectext` to output is handled differently than other `gentext`, so security classification markings are not discussed in this section. See **Secdesc** on page 224 for details.

Similarly, `idrefsep` and `idreffinalsep` output generated text. However, they are used only for supporting an IDREFS attribute and are also not covered in this section. Additional information is available on page 82.

`Gentext` includes numbering of document divisions and list items; literal text; attribute content; table of contents; list of figures; etc. — anything allowed in `usetext` source or `savetext` `contrule`, as detailed in **Table 77 Savetext contrule and usetext source syntax** on page 461.

FOSI-generated text includes sorted indexes, as discussed in **Generated sorted/merged/grouped material (indexing)** on page 139.

Unless coded otherwise, `gentext` displays in the Edit window as well as in print/PDF output.

Generated text in the Edit window

Generated text in the Edit window details placement of generated text in the Edit window with tags on.

Table 11 Generated text placement in Edit window

Tag	Placement	
	Before	After
Tag pair	Gentext appears immediately after the start tag	Gentext appears immediately before the end tag
Singleton tag	Gentext appears immediately after the singleton tag	Gentext appears immediately after any <code>placemnt="before"</code> gentext. If there is no <code>"before"</code> gentext, <code>placemnt="after"</code> gentext appears immediately after the singleton tag.

Optimizing generated text display in the Edit window

Display of generated text is controlled by the commands discussed in . When both gentext display and auto update are enabled, gentext in the Edit window is automatically updated as the user edits the document. The FOSI can be coded as follows to optimize the performance of gentext updates.

When gentext comes from counters and string variables that are used only within certain contexts, the counter and/or string variables should be reset at the beginning of the context element and again at the end of the context element, as illustrated in **Figure 74**.

Figure 74 Optimizing gentext display

FOSI fragment

```
<stringdecl textid="stepct.txt" literal="">
...
<e-i-c gi="procedure">
<charlist inherit="1" charsubsetref="block">
<reset resetlist="stepct stepct.txt">
...
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<charsubset>
<usetext placemnt="after"
<subchars>
<reset resetlist="stepct stepct.txt">
</subchars>
</usetext>
...
<e-i-c gi="step" context="procedure">
<charlist inherit="1" charsubsetref="block">
<enumerat enumid="stepct" increm="1">
<savetext textid="stepct.txt" conrule="stepct,\\.\">
<usetext placemnt="before" source="stepct.txt,@1.5pi">
...
```

NOTE: The specval test for editor-only is optional. The reset could apply to print/PDF as well. The important thing is that the reset is processed at the end of the context element (in this case, <procedure>) with placemnt="after".

Another example is shown in **Figure 259 Screen FOSI displays messages for users** on page 482.

FOSI TIP 

`<_touchup>` is not removed from generated text and can be used to specify most formatting.

FOSI TIP 

To disable or redefine `<_touchup>`, `<_nolinebreak>`, and `<_nopagebreak>` in certain contexts, create `<_touchup>` e-i-c's for those specific contexts.

Processing instructions in generated text

Processing instructions that are saved to a string variable as part of `#CONTENT` do not retain their usual function when they are output from a `usetext` source. This includes `<_touchup>` tags, which are discussed on page 549. As detailed in **Table 12** below, some `<_touchup>` tags are stripped when generated text is output.

NOTE: Touchup tags in the document are not affected by `gentext` processing.

Table 12 Processing instructions in generated text

PI	Gentext
<code><_font></code>	removed
<code><_hardspace></code>	softened†
<code><_hyphen-point></code>	not removed
<code><_newline></code>	removed ‡
<code>_nolinebreak</code>	removed
<code><_newpage></code>	removed
<code><_newcolumn></code>	removed
<code><_kern></code>	not removed
<code><_eos_space></code>	not removed
<code><_touchup></code>	not removed
<code><_nolinebreak></code>	not removed
<code><_nopagebreak></code>	not removed
<code><_texmac></code>	not removed
<code><_texmacpair></code>	not removed
<code><_interword-space></code>	not removed

†`<_hardspace>` is converted into a regular space in generated text

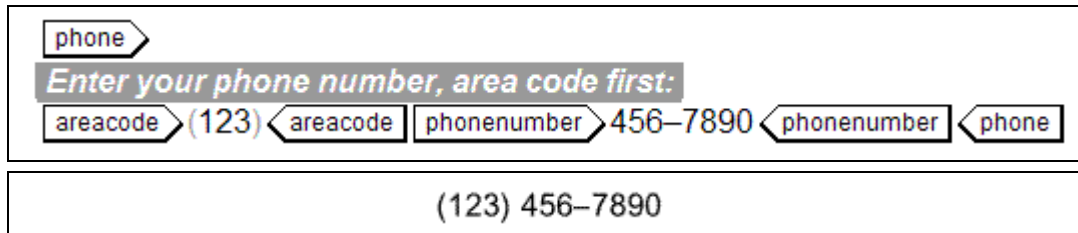
‡`<_hyphen-point>` takes effect whether hyphenation is enabled in the FOSI or not

Also see **FOSI and processing instructions** on page 543.

Generated text examples

The following figure illustrates gentext placement for a tag pair. The first graphic shows the Edit window display, and the second graphic shows the composed print/PDF output. As shown in the FOSI fragment, attribute rules are used to specify different gentext and formatting for the Editor display and composed output.

Figure 75 Gentext before and after tag pair in Edit window



FOSI fragment

```
<e-i-c gi="areacode" context="phone">
<charlist inherit="1"></charlist>
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<usetext source="\(\\" placemnt="before">
<subchars>
<highlt inherit="1" fontclr="#999999">
</subchars>
</usetext>
<usetext source="\)\\" placemnt="after">
<subchars>
<highlt inherit="1" fontclr="#999999">
...
<att>
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<usetext source="\(\\" placemnt="before"></usetext>
<usetext source="\)\\" placemnt="after"></usetext>
...
<e-i-c gi="phone">
<charlist inherit="1" charsubsetref="block"></charlist>
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<usetext source="\ Enter your phone number, area code first: \"
placemnt="before">
<subchars charsubsetref="block bold italic">
<highlt inherit="1" bckclr="#999999" fontclr="#FFFFFF">
```

```

...
<e-i-c gi="phonenumber" context="phone">
<charlist inherit="1"></charlist>
...

```

In the next example, the <date> element's attributes are tested with special categories. When the attribute values are incorrect or questionable, a message is displayed in the Edit window. Different gentext is output for print/PDF.

Figure 76 Gentext before and after singleton

date day="31" month="1" year="2008"	
date day="29" month="2" year="2008"	<i>Day is >28 -- is this a Leap Year?</i>
date day="31" month="3" year="2008"	
date day="31" month="4" year="2008"	<i>Thirty days hath April</i>
date day="31" month="5" year="2008"	
date day="31" month="6" year="2008"	<i>Thirty days hath June</i>
date day="31" month="7" year="2008"	
date day="31" month="8" year="2008"	
date day="31" month="9" year="2008"	<i>Thirty days hath September</i>
date day="31" month="10" year="2008"	
date day="31" month="11" year="2008"	
date day="31" month="12" year="2008"	

January 31, 2008	
— POSSIBLE ERROR CONDITION —	
February 29, 2008	
March 31, 2008	
— ERROR CONDITION —	
April 31, 2008	
May 31, 2008	
— ERROR CONDITION —	
June 31, 2008	
July 31, 2008	
August 31, 2008	
— ERROR CONDITION —	
September 31, 2008	
October 31, 2008	
November 31, 2008	
December 31, 2008	

DTD fragment

```
<!ELEMENT date EMPTY>
```

```
<!ATTLIST date day NUMBER #REQUIRED
           month NUMBER #REQUIReD
           year NUMBER #REQUIRED>
```

FOSI fragment

```
<stringdecl textid="day.txt" literal="">
<stringdecl textid="month.txt" literal="">
<stringdecl textid="month1.txt" literal="January">
<stringdecl textid="month2.txt" literal="February">
<stringdecl textid="month3.txt" literal="March">
<stringdecl textid="month4.txt" literal="April">
<stringdecl textid="month5.txt" literal="May">
<stringdecl textid="month6.txt" literal="June">
<stringdecl textid="month7.txt" literal="July">
<stringdecl textid="month8.txt" literal="August">
<stringdecl textid="month9.txt" literal="September">
<stringdecl textid="month10.txt" literal="October">
<stringdecl textid="month11.txt" literal="November">
<stringdecl textid="month12.txt" literal="Decenber">
<stringdecl textid="year.txt" literal="">
...
<charsubset charsubsetid="error-condition">
<usetext source="\ - ERROR CONDITION - \" placemnt="before">
<subchars charsubsetref="block"></subchars>
...
<charsubset charsubsetid="author-guidelines"
charsubsetref="endline bold italic">
<highlt inherit="1" bckclr="#999999" fontclr="#FFFFFF">
...
<e-i-c gi="date">
<charlist inherit="1" charsubsetref="startline"></charlist>
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="2">
<specval attname="day" attloc="date" attval="#GT#28">
<charsubset>
<usetext source="\ Day is >28 -- is this a Leap Year? \" placemnt="before">
<subchars charsubsetref="author-guidelines"></subchars>
...
<att logic="and">
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="2">
<specval attname="day" attloc="date" attval="#GT#28">
<charsubset>
<usetext source="\ - POSSIBLE ERROR CONDITION - \" placemnt="before">
<subchars charsubsetref="block"></subchars>
...
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="4">
<specval attname="day" attloc="date" attval="#GT#30">
```

```

<charsubset>
<usetext source="\ Thirty days hath April \" placemnt="before">
<subchars charsubsetref="author-guidelines"></subchars>
...
<att logic="and">
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="4">
<specval attname="day" attloc="date" attval="#GT#30">
<charsubset charsubsetref="error-condition"></charsubset>
</att>
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="6">
<specval attname="day" attloc="date" attval="#GT#30">
<charsubset>
<usetext source="\ Thirty days hath June \" placemnt="before">
<subchars charsubsetref="author-guidelines"></subchars>
...
<att logic="and">
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="6">
<specval attname="day" attloc="date" attval="#GT#30">
<charsubset charsubsetref="error-condition"></charsubset>
</att>
<att logic="and">
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="9">
<specval attname="day" attloc="date" attval="#GT#30">
<charsubset>
<usetext source="\ Thirty days hath September \" placemnt="before">
<subchars charsubsetref="author-guidelines"></subchars>
...
<att logic="and">
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<specval attname="month" attloc="date" attval="9">
<specval attname="day" attloc="date" attval="#GT#30">
<charsubset charsubsetref="error-condition"></charsubset>
</att>
<att>
<fillval attname="day" attloc="date" fillcat="savetext" fillchar="conrule">
<charsubset>
<savetext textid="day.txt">
...
<att>
<specval attname="month" attloc="date" attval="1">
<charsubset>
<savetext textid="month.txt" conrule="month1.txt">
...
<att>
<specval attname="month" attloc="date" attval="2">
<charsubset>

```

```
<savetext textid="month.txt" conrule="month2.txt">
...
<att>
<specval attname="month" attloc="date" attval="3">
<charsubset>
<savetext textid="month.txt" conrule="month3.txt">
...
<att>
<specval attname="month" attloc="date" attval="4">
<charsubset>
<savetext textid="month.txt" conrule="month4.txt">
...
<att>
<specval attname="month" attloc="date" attval="5">
<charsubset>
<savetext textid="month.txt" conrule="month5.txt">
...
<att>
<specval attname="month" attloc="date" attval="6">
<charsubset>
<savetext textid="month.txt" conrule="month6.txt">
...
<att>
<specval attname="month" attloc="date" attval="7">
<charsubset>
<savetext textid="month.txt" conrule="month7.txt">
...
<att>
<specval attname="month" attloc="date" attval="8">
<charsubset>
<savetext textid="month.txt" conrule="month8.txt">
...
<att>
<specval attname="month" attloc="date" attval="9">
<charsubset>
<savetext textid="month.txt" conrule="month9.txt">
...
<att>
<specval attname="month" attloc="date" attval="10">
<charsubset>
<savetext textid="month.txt" conrule="month10.txt">
...
<att>
<specval attname="month" attloc="date" attval="11">
<charsubset>
<savetext textid="month.txt" conrule="month11.txt">
...
<att>
<specval attname="month" attloc="date" attval="12">
<charsubset>
<savetext textid="month.txt" conrule="month12.txt">
```

```

...
<att>
<fillval attname="year" attloc="date" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext textid="year.txt">
...
<att>
<specval attname="print-only" attloc="SYSTEM-VAR" attval="yes">
<charsubset>
<usetext source="month.txt,\ \,day.txt,\, \,year.txt" placemnt="after">
<subchars charsubsetref="block"></subchars>
...

```

Gentext placement for a singleton that is declared in the document type configuration file (DCF) as a graphic is illustrated in the following figures.

Figure 77 Gentext before graphic in Edit window



FOSI fragment

```

<e-i-c gi="graphic">
<charlist inherit="1">
<usetext source="\Side view: \" placemnt="before"></usetext>
...

```

Figure 78 Gentext after graphic in Edit window



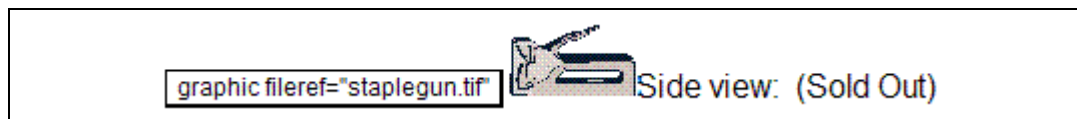
FOSI fragment

```

<e-i-c gi="graphic">
<charlist inherit="1">
<usetext source="\ (Sold Out)\" placemnt="after"></usetext>
...

```

Figure 79 Gentext before and after graphic in Edit window



FOSI fragment

```
<e-i-c gi="graphic">
<charlist inherit="1">
<usetext source="\Side view: \" placemnt="before"></usetext>
<usetext source="\ (Sold Out)\\" placemnt="after" ></usetext>
...
```

Figure 80 Gentext before and after graphic in print/PDF**FOSI fragment**

```
<e-i-c gi="graphic">
<charlist inherit="1">
<usetext source="\Side view: \" placemnt="before"></usetext>
<usetext source="\ (Sold Out)\\" placemnt="after" ></usetext>
...
```

Generated tables

A generated table (**gentable**) is a table that is created by the FOSI from elements that are not table elements. A gentable is output from a `usetext` source, using the exclamation point (!) delimiter, which is discussed on page 74.

NOTE: Gentables display in the Edit window as well as in print/PDF output.

NOTE: Don't use gentables when `algroup` could easily be used instead. See **Table limitations** on page 537 for details.

Generated tables examples

The first example shows a one-cell FOSI-generated table that appears as a shaded box in the Edit window. A gentable is used because boxing is not supported for Edit window display. The delimiters in the `savetext` `conrule` and `usetext` source are highlighted.

FOSI Tip 

Set `gentexttagdisplay=full` displays gentext processing instructions and FOSI pseudo-elements with generated text in the Edit window. You can start a style panel for a pseudo-element by clicking after it in the Edit window and entering `eic` at the command line.

FOSI Tip 

The easiest way to develop gentable code is to create a table in Arbortext Editor with the desired formatting. Then use an ASCII editor to copy the markup from the `.sgm/.xml` file to the `.fos` file.

FOSI Tip 

When working with/debugging a FOSI-generated table, let the gentable appear in the Edit window as well as in print. `set gentexttagdisplay=full` at the command line displays gentext tags in the Edit window. You can click after a gentext tag and start a style panel for it.

When the gentable is coded, tested, and debugged, find out if users need or want to see the gentable. It may make sense to code the FOSI so the gentable is output only in print/PDF, not in the Edit window.

Figure 81 Shaded box (gentable) in Edit window**FOSI fragment**

```
<usetext source='!<table frame="all">
<tgroup cols="1"><colspec colname="col1" colwidth="6pi"></colspec>
<tbody><row><entry align="center" valign="bottom">
<?Pub _cellfont FamName="Rockwell Extra Bold" Shading="#CCCCCC>SHADED BOX
IN EDIT WINDOW</entry></row></tbody></tgroup></table>!'>
</usetext>
```

The next example shows a gentable that is assembled from non-table markup. It includes table formatting and header rows, including a spanned row. This example shows the use of line breaks to keep track of delimiters and avoid syntax errors. Also notice the e-i-cs for table markup with <partlist> as context.

Figure 82 Generated table of parts information

PARTS LIST		
Part Number	Description	Unit Price
12345-IOU	Hammer	\$1.23
O-689045	Saw	\$56.78
I-0901098	Screwdriver set	\$555.55

DTD fragment

```
<!ELEMENT partlist (part)+ >
<!ELEMENT part (graphic, partno, partdesc, unit-price) >
<!ELEMENT partno | partdesc | unit-price) (#PCDATA) >
<!ATTLIST graphic fileref CDATA #IMPLIED >
```

XML fragment

```
<partlist>
<part>
<partno>12345</partno>
<partdesc>Hammer</partdesc>
<unit-price>12.34</unit-price>
</part>
<part>
```

```

<partno>0-689</partno>
<partdesc>Saw</partdesc>
<unit-price>56.78</unit-price>
</part>
<part>
<partno>I-0901</partno>
<partdesc>Screwdriver set</partdesc>
<unit-price>55.55</unit-price>
</part>
</partlist>

```

FOSI fragment

```

<stringdecl textid="parts.app">
<stringdecl textid="partno.txt" literal="">
<stringdecl textid="partdesc.txt" literal="">
<stringdecl textid="unit-price.txt" literal="">
...
<e-i-c gi="partno" context="part partlist">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="partno.txt" conrule="#CONTENT">
...
<e-i-c gi="partdesc" context="part partlist">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="partdesc.txt" conrule="#CONTENT">
...
<e-i-c gi="unit-price" context="part partlist">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="unit-price.txt" conrule="#CONTENT">
...
<e-i-c gi="part" context="partlist">
<charlist inherit="1" charsubsetref="block">
<savetext textid="parts.app" placemnt="after" append="1"
conrule='!<row><entry colname="col1" colsep="1" align="left">!
,partno.txt,
!</entry><entry colname="col2" colsep="1" align="left">!
,partdesc.txt,
!</entry><entry colname="col3" colsep="1" align="char" char="."
charoff="50">!
,\$,unit-price.txt,
!</entry></row>!>'>
...
<e-i-c gi="partlist">
<charlist inherit="1" charsubsetref="block">
<usetext placemnt="after" source='!<table frame="all"><tgroup cols="3">
<colspec colname="col1" colwidth="1in"/>
<colspec colname="col2" colwidth="1.5in"/>
<colspec colname="col3" colwidth="1in"/>
<thead><row>
<entry namest="col1" nameend="col3" align="center">
<?Pub_cellfont Shading="#888888"?>PARTS LIST</entry>
</row><row>

```

FOSI Tip 

Graphic entities cannot be declared in the FOSI style panels interface. Only the tagged FOSI editor can be used.

```

<entry colname="col1" colsep="1" align="center">
<?Pub _cellfont Shading="#888888"?>Part Number</entry>
<entry colname="col2" colsep="1" align="center">
<?Pub _cellfont Shading="#888888"?>Description</entry>
<entry colname="col3" colsep="1" align="center">
<?Pub _cellfont Shading="#888888"?>Unit Price</entry>
</row></thead><tbody>!
,parts.app,
!</tbody></tgroup></table>!<'>
</usetext>
...
<e-i-c gi="table" context="partlist">
<charlist inherit="1">
<font inherit="1" famname="Tahoma">
...
<e-i-c gi="tbody" context="* table partlist">
<charlist inherit="1">
...
<e-i-c gi="thead" context="* table partlist">
<charlist inherit="1" charsubsetref="bold">
<highlt inherit="1" fontclr="#FFFFFF">
...

```

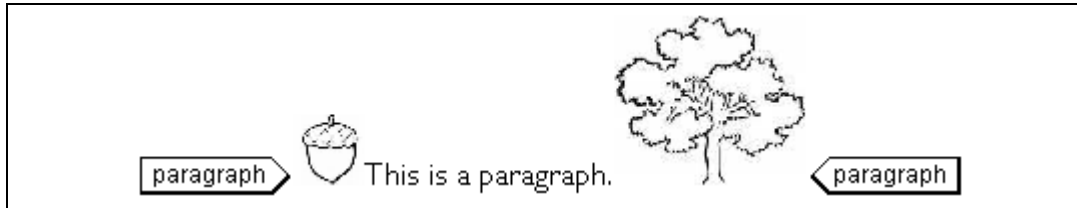
Generated graphics

A **FOSI-generated graphic** is output from the putgraph category, which references a graphic entity that has been declared in the FOSI. Generated graphics can be displayed in the Edit window as well as in print/PDF output.

In the Edit window, the generated graphic is displayed after the start tag of a tag pair when placeemnt="before". When placeemnt="after", the generated graphic appears before the end tag of a tag pair. When the tag is a singleton, the output is the same for placeemnt="before" and placeemnt="after": the graphic appears after the tag.

Generated graphic example

Figure 83 Edit window display of graphics generated from tag pair



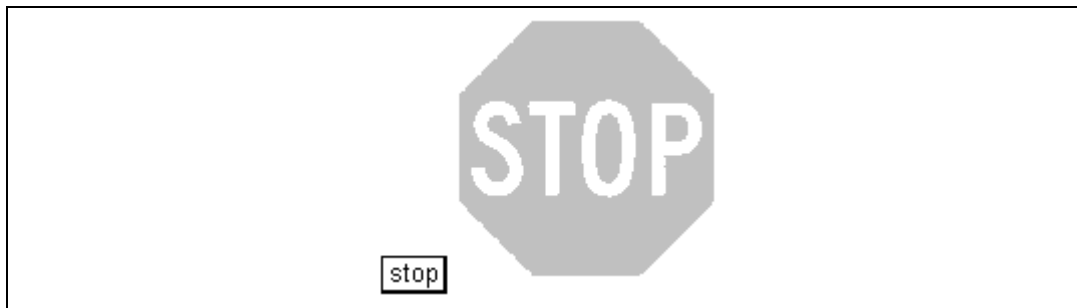
FOSI fragment

```
<!ENTITY acorn SYSTEM "acorn.jpg" NDATA jpg>
<!ENTITY oaktree SYSTEM "oaktree.jpg" NDATA jpg>
...
<e-i-c gi="paragraph">
<charlist inherit="1" charsubsetref="block pinspace">
<putgraph graphname="acorn" width="2.25pi" placemnt="before" scalefit="1">
</putgraph>
<putgraph graphname="oaktree" width="6pi" placemnt="after" scalefit="1">
...

```

When a singleton outputs a graphic, the placemnt characteristic is irrelevant. In the Edit window, graphics appear after the singleton.

Figure 84 Edit window display of graphic generated from singleton



FOSI fragment

```
<!ENTITY stop SYSTEM "stop.bmp" NDATA bmp>
...
<e-i-c gi="stop">
<charlist inherit="1">
<putgraph graphname="stop">...</putgraph>
...

```

FOSI TIP 

Use the record end character entity `&#RE;` to specify the end of a line in a generated ASCII file. Be sure to surround it with backslash (`\`) characters since character entities are literal text.

Generated files

A **FOSI-generated file** is an external ASCII file that is output from a `usetext` with `userule="1"`. All constructs allowed in `usetext` source or `savetext` `conrule`, as detailed in **Table 77 Savetext conrule and usetext source syntax** on page 461, may be exported to an external file. This includes `#CONTENT`, which may contain markup.

NOTE: When `#CONTENT` is saved to a `savetext` `conrule` and output from a `usetext` with `userule="1"`, any saved markup appears as SGML markup. For example, `Lorem ipsum <xref refid="abc"/> doned mattis enim is` written to the ASCII file without the slash (`/`).

A FOSI-generated file is named for the `e-i-c` with the `usetext`, which could be an element or a pseudo-element. This can be modified using an ACL `userule` hook.

The default location for the file is the same directory as the source document. The file extension defaults to `.exp`. These defaults can be changed by setting environment variables, as described in below.

Generated file example

In this example, the FOSI writes an external ASCII file with information related to print/PDF output.

Figure 85 External ASCII file with print report

```

Book title is Essential FOSI
Publication number is 12345-6
Total page count (not including
front and back covers) is 489
Chapter 1 starts on page 9
Chapter 2 starts on page 15
Chapter 3 starts on page 405
Chapter 4 starts on page 423

```

DTD fragment

```

<!ELEMENT book (front,body)>
<!ATTLIST book pubnumber CDATA #REQUIRED>
<!ELEMENT front (titlepage,toc)>
<!ELEMENT body (chapter+)>

```

XML fragment

```
<book pubnumber="12345-6">
<front>
<titlepage>
<title>Practical FOSI</title>
```

FOSI fragment

```
<counter initial="0" style="arabic" enumid="chapterct">
<counter initial="0" style="arabic" enumid="totalpagect">

<stringdecl textid="chapterct.txt" literal="">
<stringdecl textid="chapters.app" literal="">
<stringdecl textid="pubnumber.txt" literal="">
<stringdecl textid="totalpagect.txt" literal="">
...
<pageset id="body.page">
<rectopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...
<pagespec pgid="body.recto0">
...
<rectopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...
<pagespec pgid="body.recto1">
...
<versopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...
<pagespec pgid="body.verso1">
...
<pageset id="front.page">
<rectopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...
<pagespec pgid="front.recto0">
...
<rectopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...

```

```
<pagespec pgid="front.recto1">
...
<versopg>
<pageres>
<enumerat increm="1" enumid="totalpagect">
<savetext textid="totalpagect.txt" conrule="totalpagect">
...
<pagespec pgid="front.verso1">
...
<e-i-c gi="book">
<charlist charsubsetref="block">
<usetext placemnt="after" userule="1"
source="\Book title is \,book-title.txt,\&#RE;\,
\Publication number is \,pubnumber.txt,\&#RE;\,
\Total page count (not including front and back covers) is \,
totalpagect.txt[BO],\&#RE;\,
<chapters.app">
</usetext>
<att>
<fillval attname="pubnumber" attloc="manual" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext placemnt="before" textid="pubnumber.txt">
...
<e-i-c gi="chapter">
<charlist inherit="1" charsubsetref="block">
<textbrk startpg="recto" pageid="body.page" newpgmdl="local">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
<savetext textid="chapters.app" append="1"
conrule="\Chapter \,chapterct.txt,\ starts on page \,
totalpagect.txt[BO],\&#RE;\>
...
<e-i-c gi="front">
<charlist inherit="1" charsubsetref="block">
<textbrk startpg="recto" pageid="front.page" newpgmdl="local">
...
```

Generated HTML

You can generate structured HTML with Arbortext Editor using FOSI and an XSLT post-process.

NOTE: FOSI-generated HTML is not the same as File→Save as HTML menu selection, which uses the formatting coded in the FOSI but does not create structured HTML.

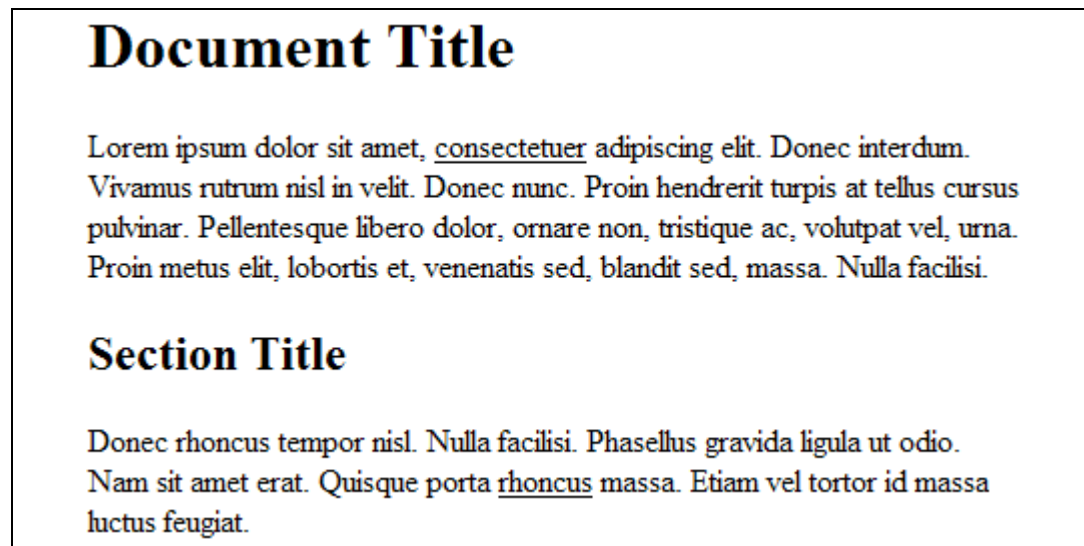
When the document is formatted or the FOSI is compiled, the FOSI generates structured HTML for block elements in the source document and writes the resulting document to an external. The file extension is `.exp` unless changed with `APTEXPORTEXT` environment variable. In the FOSI, the technique for generating tables from non-table markup described in **Generated tables** on page 129 is used to generate structured HTML. `usetext userule="1"` outputs the generated HTML to the external file.

However, the Arbortext Editor formatting engine does not transform inline elements. That is why XSLT is needed. The XSLT transformation runs quickly since the document structure is not affected.

Generated HTML example

This example shows how a simple document with two levels of titles, paragraphs, and inline elements is transformed into structured HTML. `usetext userule="1"` in the FOSI writes the appended string with the HTML tags and content to an external file. `File→Compose→XSL...` is used with the external file and the XSLT stylesheet to transform inline element names and create the final HTML file.

Figure 86 Generated structured HTML file



XML fragment

```

<document>
<title>Document Title</title>
<para>Lorem ipsum dolor sit amet, <underline>consectetuer</underline>
adipiscing elit. Donec interdum. Vivamus rutrum
nisl in velit. Donec nunc. Proin hendrerit turpis at tellus cursus
pulvinar. Pellentesque libero dolor, ornare non, tristique
ac, volutpat vel, urna. Proin metus elit, lobortis et, venenatis sed,
blandit sed, massa. Nulla facilisi.</para>
<section>
<title>Section Title</title>
<para>Donec rhoncus tempor nisl. Nulla facilisi. Phasellus gravida ligula
ut odio. Nam sit amet erat. Quisque porta <underline>rhoncus</underline>
massa. Etiam vel tortor id massa luctus feugiat.</para>
...

```

FOSI fragment

```

<stringdecl textid="outut-html.app" literal="">
...
<e-i-c gi="document">
<charlist inherit="1" charsubsetref="block">
...
<usetext source="<output-html>,</output-html>" placemnt="after">
</usetext>
...
<e-i-c gi="output-html">
<charlist inherit="1">
<usetext source='\<?xml version="1.0" encoding="UTF-8"
standalone="no"?>\,<html>,<head>,</head>,<body>,output-html.app,
</body>,</html>'
userule="1"></usetext>
...
<e-i-c gi="para">
<charlist inherit="1" charsubsetref="block pinspace">
<savetext textid="output-html.app" conrule="<p>,#CONTENT,</p>"
append="1">
...
<e-i-c gi="title" context="document">
<charlist inherit="1" charsubsetref="title-level1">
<savetext textid="output-html.app" conrule="<h1>,#CONTENT,</h1>"
append="1">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title-level2">
<savetext textid="output-html.app" conrule="<h2>,#CONTENT,</h2>"
append="1">
...
<e-i-c gi="underline">
<charlist inherit="1" charsubsetref="inline underline "></charlist>
</e-i-c>

```

Exported file fragment

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<html>
<head></head>
<body>
<h1>Document Title</h1>
<p>Lorem ipsum dolor sit amet, <u>consectetuer</u>
adipiscing elit. Donec interdum. Vivamus rutrum nisl in velit. Donec
nunc. Proin hendrerit turpis at tellus cursus pulvinar. Pellentesque
libero dolor, ornare non, tristique ac, volutpat vel, urna. Proin
metus elit, lobortis et, venenatis sed, blandit sed, massa. Nulla f
acilisi.</p>
<h2>Section Title</h2>
<p>Donec rhoncus tempor nisl. Nulla facilisi. Phasellus gravida
ligula ut odio. Nam sit amet erat. Quisque porta <smallcaps>rhoncus
</smallcaps> massa. Etiam vel tortor id massa luctus feugiat.</p>
...
```

XSLT stylesheet

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="@* | node()">
<xsl:copy>
<xsl:apply-templates select="@* | node()"></xsl:apply-templates>
</xsl:copy>
</xsl:template>
<xsl:template match="underline">
<u>
<xsl:apply-templates></xsl:apply-templates>
</u>
</xsl:template>
...
```

Generated sorted/merged/grouped material (indexing)

Arbortext Editor's support for FOSI includes sorting, merging, and grouping capabilities, which are used most often for generating indexes, but which can be used for other purposes. Indexing is discussed first.

It is worth noting that an index and a table of contents are different. A table of contents is a list of titles in the document, or section of a document, listed in page order. That is, each entry in the TOC is shown in the order in which it appears in the document or section. In an **index**, entries are not listed in page-order. Instead, index entries are listed in the order that results from

FOSI Tip 

It is overkill to use indexing to support a table of contents, and it may increase the number of formatting passes needed to format the document and display the TOC. Instead, for a TOC, use an appended time-independent string variable.

sorting, merging, and grouping according to the indexing scheme, which is usually alphabetical. Index entries may include a page number or another way to locate the entry in the document body, such as a section number.

Index entries may be arranged in levels so that related entries are grouped together. Typically, the first level is sorted alphabetically. For example, in an index with two levels, level 1 entries in an index may include “fruits” and “vegetables” while level 2 entries may include “apple” under “fruits” and “carrot” under “vegetables.” For example:

```
fruits
  apple

vegetables
  carrot
```

NOTE: Four levels of indexing are directly supported in Arbortext Editor. More levels can be used by modifying the appropriate configuration file, as described in **Default index configuration files** on page 167.

Index entries generally contain locator information for the purpose of finding the entry in the document. Typically the locators are document page numbers. In any case, locators may appear with entries at any or none of the index levels.

Document page numbers take various forms. In some documents, different numbering is used in different parts. Lowercase Roman numerals (i, ii, iii, ...) is often used to number pages in the front matter while Arabic numerals (1, 2, 3, ...) number pages in the body of the document. Compound numbers with the current chapter number and a page number that resets for each chapter (1–1, 1–2, ..., 2–1, 2–2, ...) are also commonly used to number pages in the body of a document.

Index locators do not have to be page numbers. They can be section numbers such as 1.0, 1.1,1.2, ..., 2.0, 2.1, 2.2,

NOTE: When locators in an index are not strictly digits, a parallel locator that is strictly digits is required to support sorting. See **Figure 111 Section numbers as locators** on page 185 below.

Arbortext Editor sorting/merging/grouping can be used for other purposes, such as a numerically sorted Parts List table generated from non-table markup, as shown in **Figure 116 Sorting by a subsequent level heading** on page 188.

Arbortext Editor's default sorting options are for U.S. English. However, the default language can be changed. Also, sort orders, custom symbol sort

parameters, and index group headings can be customized, as discussed in sections below.

NOTE: When indexing is used, do not code `stringdecl literal=""` in the `rsrcdesc` to initialize string variables because it results in indexing error messages and prevents the index from appearing unless `allpasses` is specified for formatting/previewing/printing. One exception: this is not an issue when formatting pass reduction is enabled.

Indexing basics

An index entry and the page on which it occurs are surrounded with **preliminary index markup** and saved to an appended string variable, such as `index.app`. The string variable is output from a `usetext` source with `userule="2"`. The `userule` enables index processing, which transforms the preliminary index markup into **final index markup**, sorts, merges, and groups the entries, and formats them according to `e-i-cs` in the FOSI for the final index markup.

NOTE: When an index is used as a table of contents at the beginning of the document, the appended string variable must be declared as a time-independent variable.

Indexing is discussed further in the sections below.

Index requirements

Answers to the questions below are needed to code an index.

- How many indexes does the document have? Be sure they are indexes, not tables of contents.
- What markup is used to add entries to the index?
- What kinds of page numbers are used in the document — simple (1, 2, 3, ...) or compound (1–1, 1–2, 1–3, ...)?
- Are “See” and “See Also” needed in the index?
- What should be the default indexing language?
- What other languages are used?
- Are there any exceptions to the index sorting order?
- How should leading punctuation be sorted?
- Should any letters be combined under a single heading, such as “XYZ”?

FOSI TIP

Indexes generally consist of text entries, but it is possible to include graphics in an index. This is left as an exercise for the reader.

- Should non-alphabetic entries be sorted separately? Should they appear before or after the alphabetic index entries?
- How should each component in the index be formatted?

DTD index markup

One or more elements for adding entries to the index can be defined in the DTD. Such elements can take several forms:

- a tag pair that surrounds the content for the index; for example:

XML fragment

```
<indexentry>structured markup</indexentry>
```

- a tag pair containing a required child element that surrounds content plus optional child elements that surround content in order to support multiple levels of indexing; for example:

XML fragment

```
<indexterm><primary>vegetables</primary><secondary>root</secondary>  
<tertiary>carrot</tertiary></indexterm>...<indexterm><primary>fruits  
</primary></indexterm>...
```

- a singleton with one or more attributes for index entries; for example:

XML fragment

```
<indxflag ref1="music" ref2="folk" ref3="Europe and America"  
ref4="Celtic traditional music"/>
```

- a tag pair that surrounds content that has one or more attributes for additional levels; for example: `<indexitem under="structured markup">XML</indexitem>`.

The content of an index entry tag pair may be suppressed from the output stream. In the following example, the contents of `<indexentry>` are not intended to be part of the document narrative and are suppressed by the FOISI:

XML fragment

```
<para>Folk<indexentry><primary>music</primary><secondary>folk</secondary>  
</indexentry> music has been defined in several ways...</para>
```

In addition, existing elements and attributes can be used to add entries to the index. The element content can be part of the text as well as an entry in the index. For example:

Or the element content can be suppressed from the output stream but added to the index. The indexing examples explore some of the possibilities.

When content with index entry is saved for output in page headers or footers, TOC, or other purpose, the index entry element is processed when the saved string is output, which can create index problems. Instead, index entry elements in such contexts such be suppressed. In the following example the contents of a <title> element are saved for output in recto page headers. To avoid problems when <title> contains <indexentry>, the content saved to the header string variable are wrapped with a pseudo-element, which provides context for an e-i-c that suppresses <indexentry>. See **Figure 113 Suppressed index entry in page headers** on page 187 below.

An element may be defined in the DTD for the purpose of outputting the index. A singleton tag is often used because all content is generated. For example:

DTD fragment

```
<!ELEMENT index EMPTY>
```

However, a tag pair could be defined, such as the following:

DTD fragment

```
<!ELEMENT index (title),...>
```

In the case of a tag pair, the `usetext` with `userule="2"` that outputs the generated index would specify `placemnt="after"`.

However, it is not always necessary to have an element specifically for the purpose of outputting an index or other sorted/merged/grouped material. For example, an index could be output from a `usetext` with `placemnt="after"` coded in the e-i-c for a <back> container element or from the top tag in the document.

Preliminary index markup

The `savetext` `conrule` for entries appended to the index string variable must conform to the following preliminary index pseudo-element markup.

SGML DTD fragment

```
<!ELEMENT ixprelim - - (ixpt | ixstart | ixend, ixkey?,
(ixsub1, ixkey1)?, (ixsub2, ixkey2)?, (ixsub3, ixkey3)?,
((ixpgstr, ixpgkey?, ixalso?) | ixsee | ixalso)*) >
```

FOSI Tip

e-i-cs for the index entry element are usually needed in various contexts, such as the following:

FOSI fragment

```
context="header.ctx"
context="toc.ctx"
context="lot.ctx"
context="lof.ctx"
context="xref.ctx"
```

The following table provides a brief description of each preliminary index pseudo-element.

NOTE: For Pinyin sorting in Chinese, one additional index element is needed to correctly format the final index. Please see **Chinese language Pinyin sorting (Chinese_mode)** on page 166 for more information.

Table 13 Preliminary index markup

Pseudo-element	Use in savetext
<ixpt>	contains the text part of the level 1 entry. If the index entry needs page ranges rather than individual pages, use <ixstart> and <ixend>
<ixstart> and <ixend>	contains the text part of the level 1 entry. They are used when a page range is forced in the index.
<ixsub1>	contains the index entry level 2 text
<ixsub2>	contains the index entry level 3 text
<ixsub3>	contains the index entry level 4 text
<ixkey>	defines an alternative sort for a given level 1 index entry; may contain text and/or entity references.
<ixkey1>	defines an alternative sort for a given level 2 index entry; may contain text and/or entity references
<ixkey2>	defines an alternative sort for a given level 3 index entry; may contain text and/or entity references
<ixkey3>	defines an alternative sort for a given level 4 index entry; may contain text and/or entity references
<ixpgkey>	contains only digits; used for the sorting process when different numbering styles are used in the same document and when <ixpgkey> contains text and/or other pseudo-elements
<ixpgstr>	contains the numbering as it should appear in the index; may contain text and possibly other pseudo-elements. If it contains nothing but a number, then it can be used for sorting as well, otherwise a purely numeric <code>ixpgkey</code> must be included for the entry for use in sorting
<ixsee>	adds pointers to other index entries; may contain text and possibly pseudo-elements
<ixalso>	adds pointers to other index entries; may contain text and possibly pseudo-elements

Preliminary index pseudo-elements may contain the following in any combination:

- #CONTENT for element content
- #CONTENT(attribute name) for the value of an attribute
- entity references
- non-indexing pseudo-elements

NOTE: When different numbering styles are used in the same document, and when <ixpgkey> contains text and/or other pseudo-elements, <ixpgkey> must be used to support sorting.

Related ACL

set showprelimuserules= {off | on}. Assuming gentext is enabled, preliminary index entries can be displayed in the Edit window by entering the following at the command line: set showprelim=on. To see the preliminary index pseudo-elements, enter the following at the command line: set gentexttagdisplay=full. The default for set showprelimuserules is off. set showprelim is an alias for set showprelimuserules.

Preliminary index examples

The following figure shows the preliminary index markup needed to include “WYSIWYG” as an entry in the index.

Figure 87 Preliminary index markup — one level index

DTD fragment

```
<!ELEMENT indexentry ANY >
```

XML fragment

```
<para><indexentry>WYSIWYG</indexentry> is last-century technology!</para>
```

FOSI fragment

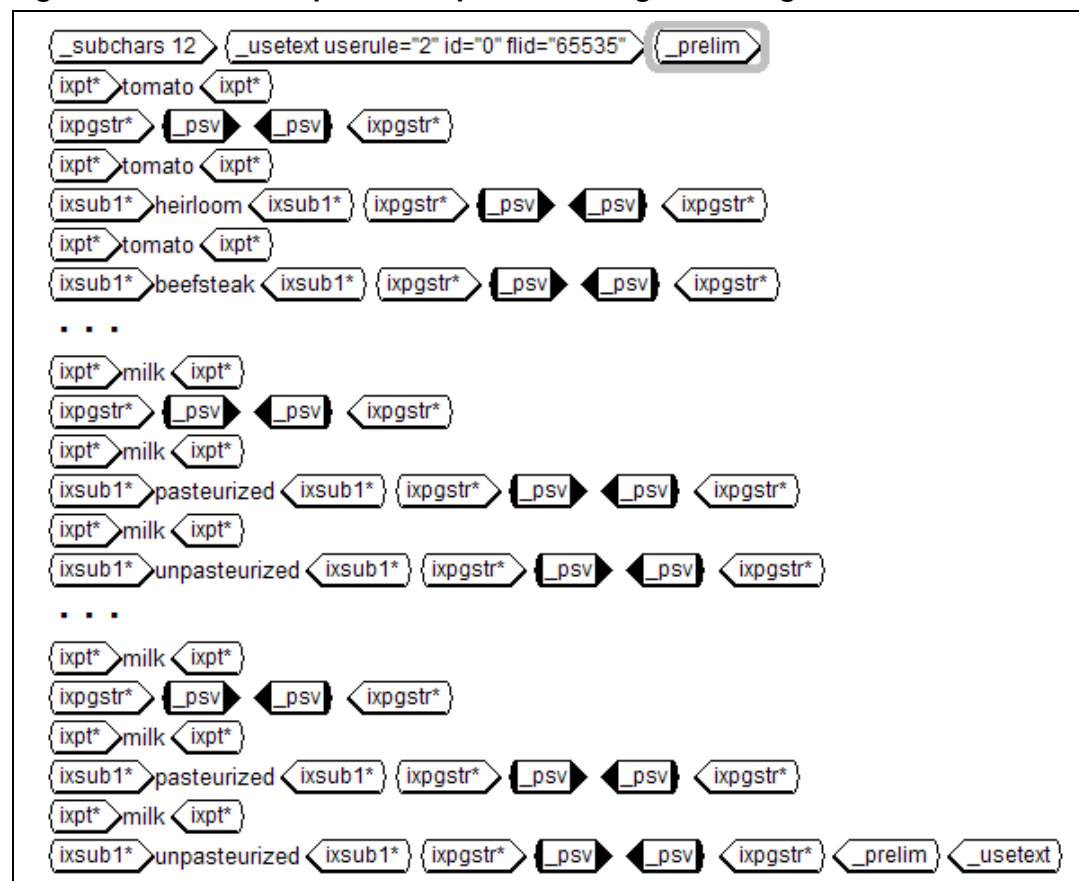
```
<stringdecl textid="folioct.txt">
<stringdecl textid="index.app" literal="">
...
<e-i-c gi="indexentry">
<charlist inherit="1 charsubsetref="inline">
...
<savetext textid="index.app" append="1"
conrule="<ixpt>, #CONTENT, </ixpt>, <ixpgstr>, folioct.txt [B0], </ixpgstr>">
...

```

The next example is a two-level index created from element and attribute content. The content of `<indexitem>` is assumed to be the level 1 entry unless the *under* attribute on `<indexitem>` is specified, in which case, the *under* attribute value becomes the level 1 entry and the element content becomes the level 2 entry.

Figure 88 shows the beginning of the preliminary index displayed in the Edit window with full gentext tags. Notice the `<_prelim>` pseudo-element that wraps the preliminary index. The `<_psv>` processing instructions in `<ixpgstr>` are placeholders for the page numbers that appear in print/PDF output. The formatting for the final index markup is shown in **Figure 93** on page 153.

Figure 88 Two-level preliminary index with gentext tags



DTD fragment

```
<!ELEMENT index EMPTY >
<!ELEMENT indexitem ANY >
<!ATTLIST indexitem under CDATA #IMPLIED >
```

XML fragment

```
<paragraph>The <indexterm>tomato</indexterm> may be an
<indexterm under="tomato">heirloom</indexterm> or a
<indexterm under="tomato">beefsteak</indexterm>...</paragraph>
<paragraph>The <indexterm>milk</indexterm> may be an
<indexterm under="milk">pasteurized</indexterm> or a
<indexterm under="milk">unpasteurized</indexterm>...</paragraph>
...
```

FOSI fragment

```
<stringdecl textid="bodyfolioct.txt">
<stringdecl textid="index.app">
...
<e-i-c gi="index">
<charlist inherit="1">
...
<usetext source="index.app" userule="2"></usetext>
...
<e-i-c gi="indexterm">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<specval attname="under" attloc="indexterm" attval="#NONE">
<charsubset>
<savetext append="1" textid="index.app"
conrule="<ixpt>, #CONTENT, </ixpt>,
<ixpgstr>, bodyfolioct.txt[BO], </ixpgstr>">
...
<att>
<specval attname="under" attloc="indexterm" attval="#ANY">
<charsubset>
<savetext append="1" textid="index.app"
conrule="<ixpt>, #CONTENT(under), </ixpt>, <ixsub1>, #CONTENT, </ixsub1>,
<ixpgstr>, bodyfolioct.txt[BO], </ixpgstr>">
...
```

The next example shows the use of `<ixstart>` and `<ixend>` to force a range of locators for level 1 in the index. The first graphic shows page numbers for locators. The second graphic uses section numbers as locators. The third graphic shows the input document in the Edit window.

TIP 

You can double-click on the page number placeholders in the preliminary index displayed in the Edit window to link to the index entry in the document. Use the popup menu to select Go Back and return to the index entry.

Figure 89 Forced chapter page range

F	V
Fruits 1–3	Vegetables 5–9
Grapefruit 3	Broccoli 5
Kumquats 1	Brussels sprouts 6
Mangoes 1	Lettuce 6
Watermelon 2	Spinach 7

FOSI fragment

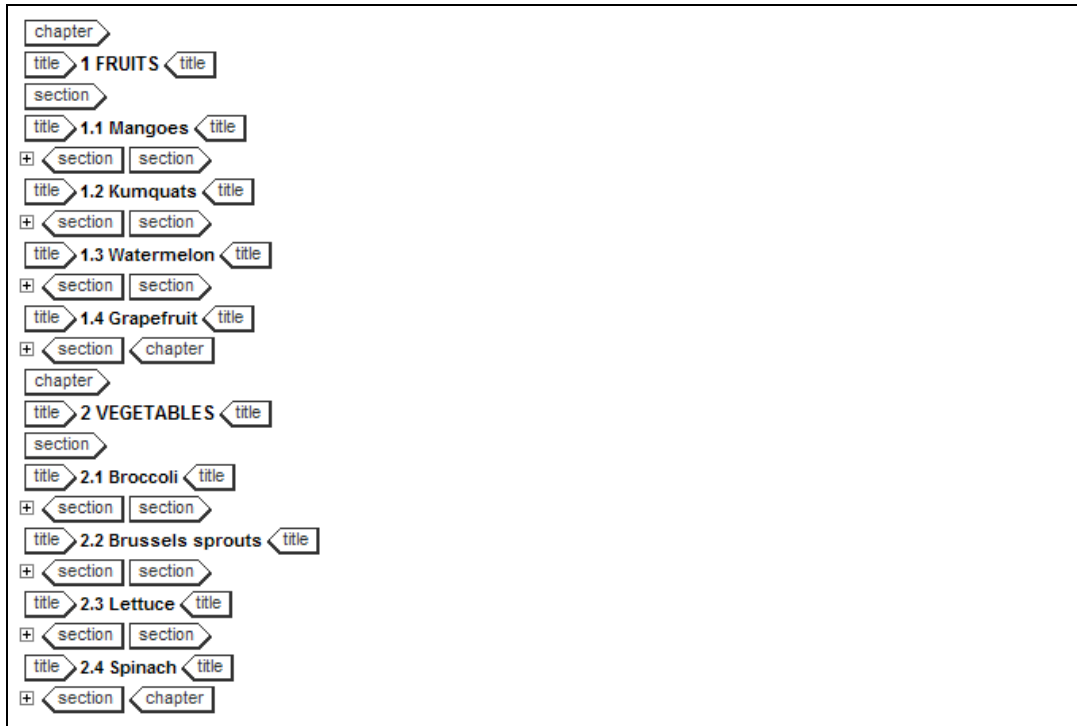
```

<stringdecl textid="bodyfoliocr.txt">
<stringdecl textid="chapter-title.txt">
<stringdecl textid="index.app">
...
<e-i-c gi="chapter">
<charlist inherit="1" charsubsetref="block">
...
<savetext textid="index.app" placemnt="after" append="1"
conrule="<ixend>,chapter-title.txt,</ixend>,"
<ixpgstr>,bodyfoliocr.txt [BO],</ixpgstr>">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
...
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<savetext textid="index.app" append="1"
conrule="<ixstart>,chapter-title.txt,</ixstart>,"
<ixpgstr>,bodyfoliocr.txt [BO],</ixpgstr>">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title italic">
...
<savetext textid="index.app" append="1"
conrule="<ixpt>,chapter-title.txt,</ixpt>,"
<ixsub1>,#CONTENT,</ixsub1>,"
<ixpgstr>,bodyfoliocr.txt [BO],</ixpgstr>">
...

```

Figure 90 Forced chapter number range

F	V
Fruits 1.0–1.4	Vegetables 2.0–2.4
Grapefruit 1.4	Broccoli 2.1
Kumquats 1.2	Brussels sprouts 2.2
Mangoes 1.1	Lettuce 2.3
Watermelon 1.3	Spinach 2.4



FOSI fragment

```

<counter initial="0" style="arabic" enumid="bodyfolioct">
<counter initial="0" style="arabic" enumid="chapterct">
<counter initial="0" style="arabic" enumid="sectionct">
<stringdecl textid="bodyfolioct.txt">
<stringdecl textid="chapterct.txt">
<stringdecl textid="chapter-title.txt">
<stringdecl textid="index.app">
<stringdecl textid="sectionct.txt">
...
<e-i-c gi="chapter">
<charlist inherit="1" charsubsetref="block">
...
<reset resetlist="sectionct">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
<savetext textid="sectionct.txt" conrule="sectionct">
<savetext textid="index.app" placemnt="after" append="1"
conrule="<ixend>,chapter-title.txt,</ixend>,"
<ixpgstr>,bodyfolioct.txt[B0],</ixpgstr>">
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block">

```

```

<enumerat increm="1" enumid="sectionct">
<savetext textid="sectionct.txt" conrule="sectionct">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
...
<savetext textid="chapter-title.txt" conrule="#CONTENT">
<savetext textid="index.app" append="1"
conrule="<ixstart>,chapter-title.txt,</ixstart>,
<ixpgstr>,bodyfolioct.txt [B0],</ixpgstr>">
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title italic">
...
<savetext textid="index.app" append="1"
conrule="<ixpt>,chapter-title.txt,</ixpt>,
<ixsub1>,#CONTENT,</ixsub1>,
<ixpgstr>,bodyfolioct.txt [B0],</ixpgstr>
...

```

It is not necessary to define e-i-cs in the FOSI to format preliminary index markup, but it may be useful, as shown in the following figures.

Figure 91 Preliminary index with formatting and tags

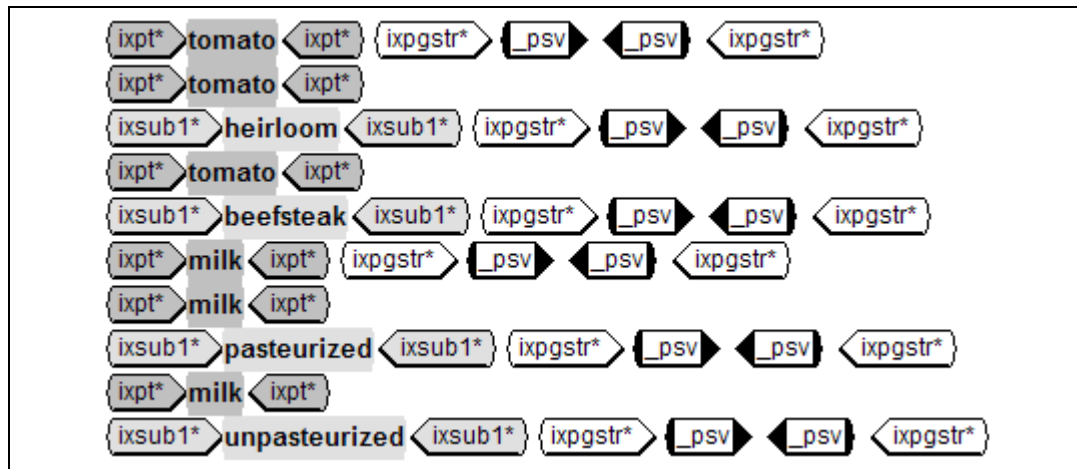


Figure 92 Preliminary index with formatting and no tags

```

tomato00
tomato
  heirloom00
tomato
  beefsteak00
milk00
milk
  pasteurized00
milk
  unpasteurized00

```

TYPE TIP 

Index entries tend to be short, so indexes are often formatted in multiple columns to save paper.

FOSI TIP 

To identify the output from final index pseudo-elements, add different colors to the e-i-cs.

Final index markup

When the Arbortext Editor formatting engine encounters `usetext userule="2"`, it sorts the entries in the preliminary index pseudo-elements and generates the index with final index markup pseudo-elements.

SGML DTD fragment

```

<!ELEMENT ixfinal - - (ixpregrp?, (ixalphagr | ixrangegrp)*,
                      ixpostgrp?) >
<!ELEMENT (ixpregrp | ixalphagr |
           ixrangegrp | ixpostgrp) - - (ixgrphd, ixentry+) >
<!ELEMENT ixgrphd - - (#PCDATA) >
<!ELEMENT ixentry - - (ixhead, (ixentry+ |((ixlocs, ixalso*) |
                      ixalso+), ixentry*) | ixsee+) >
<!ELEMENT ixlocs - - ((ixpgstr, ixendpg?)* >

```

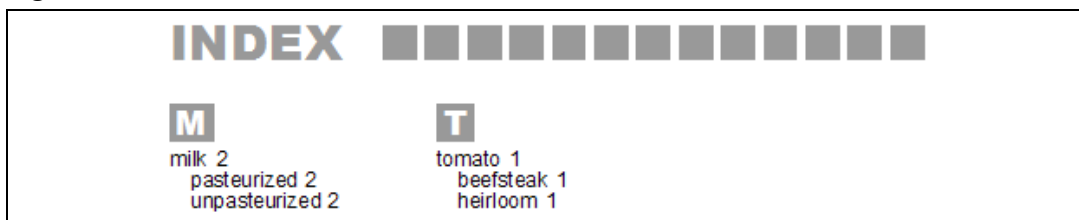
To format an index, code e-i-cs for the final index markup. The following table provides a brief description of each pseudo-element.

Table 14 Final index markup

Pseudo-element	Use in savetext
<ixgrphd>	Applies formatting to the alpha header for each <ixalphagr.>
<ixalphagr.>	Groups all primary entries for an alphabetic character or character-pair.
<ixrangegr.>	Groups all primary entries of a range of alphabetic characters.
<ixpregr.>	Groups all entries occurring before the first alphabetic entry. Use generated text to produce titles.
<ixpostgr.>	Groups all entries occurring after the last alphabetic entry. Use generated text to produce titles.
<ixhead>	Contains the formatting for the text part of individual entries.
<ixentry>	Wraps an entire entry.
<ixpgstr>	Contains formatting for page numbers in indexes. It should have <i>e-i-cs</i> for <i>first</i> and <i>notfirst</i> occurrence in case an entry is repeated on nonconsecutive pages, generating a list of page numbers in the index (the <i>notfirst</i> occurrences typically generate a comma and space before).
<ixendpg>	Contains formatting for page ranges in indexes. It is used if an entry is repeated on consecutive pages.
<ixsee>	Contains the formatting for references from an invalid index entry to a valid one. Its occurrence typically outputs a See reference.
<ixseealso>	Contains the formatting for references between valid index entries. The first occurrence typically outputs a See also reference. <i>Notfirst</i> occurrences typically generate a semicolon and space before.

Final index examples

The following figure shows the final two-level index from the DTD and XML in **Figure 88 Two-level preliminary index with gentext tags** on page 146. Notice the *e-i-c* for <ixgrphd> references the *title charsubset* with *keep-together* and *keep-next*.

Figure 93 Formatted final two-level index**FOSI fragment**

```

<charfill literal="&sqf;" padding="1pt" cfid="squarefill">
<stringdecl textid="bodyfoliact.txt">
<stringdecl textid="index.app">
...
<charsubset charsubsetid="title" charsubsetref="block left prespace
  postspace keep-together keep-next"></charsubset>
...
<e-i-c gi="index">
<charlist inherit="1" charsubsetref="new-page">
<usetext source="\INDEX\squarefill">
<subchars charsubsetref="title one-column"></subchars>
</usetext>
<usetext source="index.app" userule="2">
<subchars>
<span span="3">
...
<e-i-c gi="indexterm">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<specval attname="under" attloc="indexterm" attval="#NONE">
<charsubset>
<savetext append="1" textid="index.app"
conrule="<ixpt>, #CONTENT, </ixpt>,"
<ixpgstr>, bodyfoliact.txt[BO], </ixpgstr>">
...
<att>
<specval attname="under" attloc="indexterm" attval="#ANY">
<charsubset>
<savetext append="1" textid="index.app"
conrule="<ixpt>, #CONTENT(under), </ixpt>, <ixsub1>, #CONTENT, </ixsub1>,"
<ixpgstr>, bodyfoliact.txt[BO], </ixpgstr>">
...
<e-i-c gi="ixalphagrp">
<charlist inherit="1" charsubsetref="block"></charlist>
</e-i-c>
<e-i-c gi="ixendpg">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\-" placemnt="before"></usetext>
...

```

```

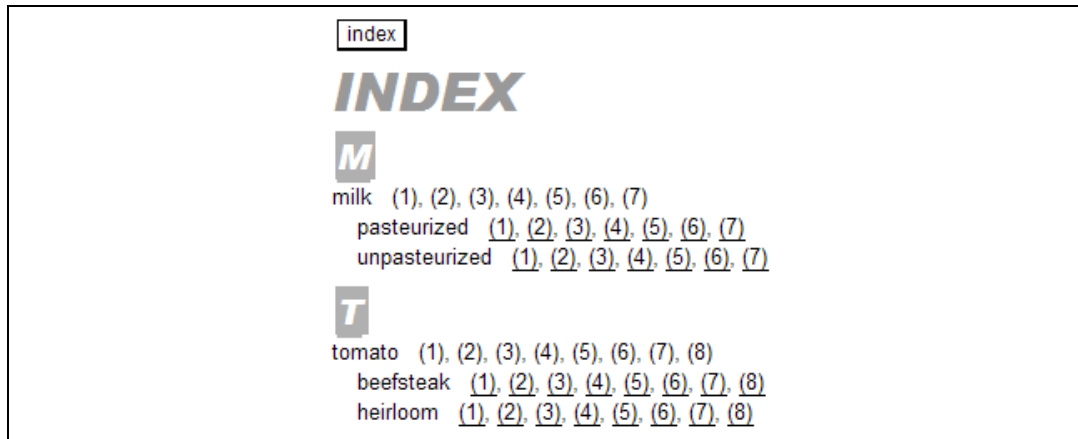
<e-i-c gi="ixentry">
<charlist inherit="1" charsubsetref="block left prespace">
<indent inherit="1" leftind="0.5em" firstln="0">
...
<e-i-c gi="ixentry" context="ixentry" occur="first">
<charlist inherit="1" charsubsetref="block left keep-previous
keep-together">
<indent leftind="1.5em" firstln="1em">
...
<e-i-c gi="ixentry" context="ixentry" occur="notfirst">
<charlist inherit="1" charsubsetref="block">
<indent leftind="1.5em" firstln="1em">
...
<e-i-c gi="ixfinal">
<charlist inherit="1" charsubsetref="block"></charlist>
...
<e-i-c gi="ixgrphd">
<charlist inherit="1" charsubsetref="title allcaps">
<font inherit="1" size="16pt">
<leading inherit="1" lead="16pt">
<indent leftind="3pt" rightind="0" firstln="*">
<highlt inherit="1" fontclr="#FFFFFF">
<presp minimum="1em" nominal="1em" maximum="1em" condit="discard"
priority="med">
<postsp minimum="5pt" nominal="5pt" maximum="5pt" priority="high">
<boxing toffset="3pt" boffset="3pt" loffset="3pt" roffset="3pt" trel="top"
brel="bottom" siderel="content" ttype="tsingle" btype="bsingle"
ltype="lsingle" rtype="rsingle" inclr="#999999" outclr="#999999">
...
<e-i-c gi="ixhead" context="ixentry">
<charlist inherit="1" charsubsetref="inline"></charlist>
...
<e-i-c gi="ixhead" context="ixentry * ixentry">
<charlist inherit="1" charsubsetref="inline"></charlist>
...
<e-i-c gi="ixlocs" context="ixentry">
<charlist inherit="1"></charlist>
...
<e-i-c gi="ixpgstr" context="* ixentry" occur="first">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\ \" placemnt="before"></usetext>
...
<e-i-c gi="ixpgstr" context="* ixentry" occur="notfirst">
<charlist inherit="1" charsubsetref="inline">
<usetext source=", \" placemnt="before"></usetext>
...

```

The following figure shows the Edit window display of the final index in **Figure 93 Formatted final two-level index** on page 153.

NOTE: The underlined numbers shown in the final index in the Edit window are not page numbers. They are links to the index entry in the document.

Figure 94 Final index in Edit window



Final index pseudo-elements can be viewed in the Edit window, which is useful for debugging. This example shows the same index as in **Figure 94 Final index in Edit window** on page 155, with some Edit window-specific coding for `<ixgrphd>` in order to display the alphabetical group heads in the Editor. Notice the `<_final>` pseudo-element that wraps the final index.

TIP 

You can double-click on the page number placeholders in the final index displayed in the Edit window to link to the index entry in the document. Use the popup menu to select Go Back and return to the index entry.

Figure 95 Final index markup in Edit window

_subchars 12 _usetext userule="2" id="0" flid="65535" _final
 ixalphagr*
 ixgrphd* **M** ixgrphd*
 ixentry* ixhead* milk ixhead* ixlocs* ixpgstr* (1) ixpgstr* ixpgstr* (2) ixpgstr*
 ixpgstr* (3) ixpgstr* ixpgstr* (4) ixpgstr* ixpgstr* (5) ixpgstr* ixpgstr* (6)
 ixpgstr* ixpgstr* (7) ixpgstr* ixlocs*
 ixentry* ixhead* pasteurized ixhead* ixlocs* ixpgstr* _gtlink (1) _gtlink
 ixpgstr* ixpgstr* _gtlink (2) _gtlink ixpgstr* ixpgstr* _gtlink (3) _gtlink
 ixpgstr* ixpgstr* _gtlink (4) _gtlink ixpgstr* ixpgstr* _gtlink (5) _gtlink
 ixpgstr* ixpgstr* _gtlink (6) _gtlink ixpgstr* ixpgstr* _gtlink (7) _gtlink
 ixpgstr* ixlocs* ixentry*
 ixentry* ixhead* unpasteurized ixhead* ixlocs* ixpgstr* _gtlink (1) _gtlink
 ixpgstr* ixpgstr* _gtlink (2) _gtlink ixpgstr* ixpgstr* _gtlink (3) _gtlink
 ixpgstr* ixpgstr* _gtlink (4) _gtlink ixpgstr* ixpgstr* _gtlink (5) _gtlink
 ixpgstr* ixpgstr* _gtlink (6) _gtlink ixpgstr* ixpgstr* _gtlink (7) _gtlink
 ixpgstr* ixlocs* ixentry*
 ixentry*
 ixalphagr*

ixalphagr*
 ixgrphd* **T** ixgrphd*
 ixentry* ixhead* tomato ixhead* ixlocs* ixpgstr* (1) ixpgstr* ixpgstr* (2)
 ixpgstr* ixpgstr* (3) ixpgstr* ixpgstr* (4) ixpgstr* ixpgstr* (5) ixpgstr*
 ixpgstr* (6) ixpgstr* ixpgstr* (7) ixpgstr* ixpgstr* (8) ixpgstr* ixlocs*
 ixentry* ixhead* beefsteak ixhead* ixlocs* ixpgstr* _gtlink (1) _gtlink
 ixpgstr* ixpgstr* _gtlink (2) _gtlink ixpgstr* ixpgstr* _gtlink (3) _gtlink
 ixpgstr* ixpgstr* _gtlink (4) _gtlink ixpgstr* ixpgstr* _gtlink (5) _gtlink
 ixpgstr* ixpgstr* _gtlink (6) _gtlink ixpgstr* ixpgstr* _gtlink (7) _gtlink
 ixpgstr* ixpgstr* _gtlink (8) _gtlink ixpgstr* ixlocs* ixentry*
 ixentry* ixhead* heirloom ixhead* ixlocs* ixpgstr* _gtlink (1) _gtlink
 ixpgstr* ixpgstr* _gtlink (2) _gtlink ixpgstr* ixpgstr* _gtlink (3) _gtlink
 ixpgstr* ixpgstr* _gtlink (4) _gtlink ixpgstr* ixpgstr* _gtlink (5) _gtlink
 ixpgstr* ixpgstr* _gtlink (6) _gtlink ixpgstr* ixpgstr* _gtlink (7) _gtlink
 ixpgstr* ixpgstr* _gtlink (8) _gtlink ixpgstr* ixlocs* ixentry*
 ixentry*
 ixalphagr*
 _final _usetext _subchars

FOSI fragment

```

<e-i-c gi="ixgrphd">
<charlist inherit="1" charsubsetref="title allcaps">...</charlist>
<att>
<specval attname="editor-only" attloc="SYSTEM-VAR" attval="#ANY">
<charsubset>
<highlt inherit="1" bckclr="gray5">
...

```

Exporting indexes for external processing

The contents of the string variable to which index entries are appended can be exported to an external file. The `savetext` for index entries is coded as usual. The `usetext` that outputs the appended variable is coded with `userule="1"` instead of `userule="2"`. When the document is formatted and all page numbering are resolved, the index with preliminary index markup but final page numbers is output to an external file named for the `e-i-c` with the `usetext` plus a `.exp` extension.

More information on **Generated files** is located on page 134.

The preliminary index markup and the index entries can be processed by ACL and other scripting, with the results returned to Arbortext Editor for final index processing. Refer to .

Exported index example

It may turn out that some index entries start with an uppercase letter while others start with a lowercase letter. The graphic in the first figure shows the final index output in such cases, with separate upper- and lowercase entries. **Figure 97** shows the contents of the `.exp` file generated by `userule="1"` in **Figure 96**. The `.exp` file can be processed by ACL and other scripting, and final index processing can be invoked on the resulting preliminary index markup. In this example, ACL (not shown) would be used to change index entries to mixed case/sentence case. **Figure 98 Index entries start with capital letter** on page 160 shows the output after the ACL processing.

Also refer to .

Figure 96 Index entries start with upper- and lowercase letters

P	T
publishing print 1, 3, 7 Print 1 Publishing print 1	typesetters 1, 3, 7 Typesetters 1

XML fragment

```
<paragraph>...<indexterm><primary>publishing</primary><secondary>print
</secondary></indexterm>...</paragraph>
<paragraph>...<indexterm><primary>Typesetters</primary></indexterm>...
</paragraph>
...
<paragraph>...<indexterm><primary>Publishing</primary><secondary>print
</secondary></indexterm>...</paragraph>
<paragraph>...<indexterm><primary>typesetters</primary></indexterm>...
</paragraph>
...
<paragraph>...<indexterm><primary>publishing</primary><secondary>Print
</secondary></indexterm>...</paragraph>
<paragraph>...<indexterm><primary>typesetters</primary></indexterm>...
</paragraph>
...
```

FOSI fragment

```
<counter initial="0" style="arabic" enumid="bodyfolioct">
<stringdecl textid="bodyfolioct.txt">
<stringdecl textid="idxpageref.txt">
<stringdecl textid="index.app" status="1">
<stringdecl textid="primary.txt">
<stringdecl textid="secondary.txt">
...
<pageset id="index.page">
<rectopg>
<pageres>
<enumerat increm="1" enumid="bodyfolioct">
<savetext textid="bodyfolioct.txt" conrule="bodyfolioct">
...
<e-i-c gi="index">
<charlist inherit="1">
<textbrk startpg="recto" pageid="index.page" newpgmdl="local">
<usetext source="\INDEX\ ">...</usetext>
<usetext source="index.app" userule="1">
...
<e-i-c gi="indexterm">
<charlist inherit="1" charsubsetref="inline">
<savetext textid="primary.txt" append="0" placemnt="before"
```

```

conrule="\">
<savetext textid="secondary.txt" append="0" placemnt="before"
conrule="\">
<savetext textid="idxpageref.txt" placemnt="before" append="0"
conrule="<ixpgstr>,bodyfolioct.txt[BO],</ixpgstr>">
<savetext textid="index.app" placemnt="after" append="1"
conrule="primary.txt,secondary.txt,idxpageref.txt">
...
<e-i-c gi="primary" context="indexterm">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="primary.txt" append="0"
conrule="<ixpt>, #CONTENT, </ixpt>">
...
<e-i-c gi="secondary" context="indexterm">
<charlist inherit="1" charsubsetref="SUPPRESS">
<savetext textid="secondary.txt" append="0"
conrule="<ixsub1>, #CONTENT, </ixsub1>">
...

```

Figure 97 Exported index with preliminary index markup and final page numbers

.exp file

```

<ixpt>publishing</ixpt>
<ixsub1>print </ixsub1><ixpgstr>1</ixpgstr>
<ixpt>Typesetters</ixpt>
<ixpgstr>1</ixpgstr>
<ixpt>Publishing</ixpt>
<ixsub1>print </ixsub1><ixpgstr>1</ixpgstr>
<ixpt>typesetters</ixpt>
<ixpgstr>1</ixpgstr>
<ixpt>publishing</ixpt>
<ixsub1>Print </ixsub1><ixpgstr>1</ixpgstr>
<ixpt>typesetters</ixpt>
<ixpgstr>1</ixpgstr>
<ixpt>publishing</ixpt>
<ixsub1>print</ixsub1><ixpgstr>1</ixpgstr>
<ixpt>typesetters</ixpt>
<ixpgstr>1</ixpgstr>
<ixpt>publishing</ixpt>
<ixsub1>print</ixsub1><ixpgstr>3</ixpgstr>
<ixpt>typesetters</ixpt>
<ixpgstr>3</ixpgstr>
<ixpt>publishing</ixpt>
<ixsub1>print</ixsub1><ixpgstr>7</ixpgstr>
<ixpt>typesetters</ixpt>
<ixpgstr>7</ixpgstr>

```

Figure 98 Index entries start with capital letter

P Publishing Print 1, 3, 7	T Typesetters 1, 3, 7
---	---------------------------------

useparam settings

Arbortext Editor sorting/indexing capabilities can be customized using the useparam characteristic of the usertext category that outputs the index string variable. For example:

```
<usertext source="index.app" userule="2" useparam="language=French">
```

NOTE: If more than one value is entered for useparam, they must be separated by a space. For example:

```
<usertext source="index.app" userule="2" useparam="index_levels=6  
mergeinhibit=-4">
```

Useparam settings are detailed below.

More than four index levels (index_levels)

Arbortext Editor supports up to four levels of indexing by default. However the index_levels useparam can be used to increase or decrease the number of levels. For example:

```
<usertext source="index.app" userule="2" useparam="index_levels=5">
```

The index_levels useparam changes the content model of the preliminary indexing pseudo-elements. A setting of 5 adds an additional level to the preliminary markup:

```
<!ELEMENT ixprelim - - (ixpt | ixstart | ixend,  
  ixkey?, (ixsub1, ixkey1?)?, (ixsub2, ixkey2?)?,  
  (ixsub3, ixkey3?)?, (ixsub4, ixkey4?)?,  
  ((ixpgstr, ixpgkey?, ixalso?) | ixsee | ixalso))*>
```

NOTE: The number of index levels used is subject to memory, virtual memory, and performance limitations.

Page number merging (maxunmerge)

Consecutive page numbers in an index are typically merged into a range instead of appearing separately. For example, “Hammer 2–4” is usually preferred to “Hammer 2, 3, 4.”

Page number merging is controlled by the `maxunmerge` useparam characteristic of the `usetext` category that outputs the index string variable. The value entered specifies how many page references should not be merged. Any nonzero value specifies the maximum number of consecutive page numbers to be left unmerged. For example:

```
<usetext source="index.app" userule=2" useparam="maxunmerge=1">
```

The default is 1, and does not need to be specified. It causes two or more consecutive page numbers to be merged. For example, unmerged “Saw 6, 7” is merged to “Saw 6–7.”

When `maxunmerge` is set to 2, two consecutive page numbers are not merged, but three or more are merged. For example: “Saw 6, 7” is not merged, but “Saw 6, 7, 8” is merged to “Saw 6–8.”

Setting `maxunmerge` to 0 (zero), as shown below, turns off page number merging.

```
<usetext source="index.app" userule=2" useparam="maxunmerge=0">
```

When merging is disabled, all page numbers are always listed individually.

NOTE: Ranges created by using `ixstart` and `ixend` are not affected by the `maxunmerge` useparam. See XREF Forcing page ranges using `<ixstart>` and `<ixend>` for information.

Entry merging (mergeinhibit)

```
v
vegetables
  flower bud
  broccoli
  cauliflower
leaves
  spinach
  kale
roots
  carrots
  radishes
```

Setting `mergeinhibit` to the maximum number of levels allowed (default is 4) or to any negative value disables page number merging. Any `maxunmerge` `useparam` setting has no effect.

Empty headings in entries (allowemptyheadings)

When an index entry includes more than one level and an entry does not exist for any of the higher level entries, the formatter ignores subsequent index entry levels and reports the following error message. “[A12535] ERROR (indexing): Empty top level heading; entire record will be ignored.”

For example, when a document with the following XML fragment is formatted, an error is reported and `folk` does not appear in the index.

XML fragment

```
<indexentry><primary></primary></secondary>folk</secondary></indexentry>
```

To suppress the error messages and allow such entries to appear in index, code `useparam="allowemptyheadings=1"` in the `usetext` with `userule="2"`.

NOTE: In some applications, it is desirable to allow empty headings but still generate an error message when appropriate. See for a FOSI technique that supports this.

Word spaces in index entries (keep_all_space, keep_leading_space, keep_trailing_space, keep_embedded_space)

Word spaces may be accidentally entered within index entries, which can result in incorrect sorting. Consequently, Arbortext Editor index processing strips leading and trailing spaces and reduces multiple spaces between words to a single space before sorting. However, the four `useparam` settings listed below can be used individually and in combination to override the default behavior.

- `keep_all_space`
- `keep_leading_space`
- `keep_trailing_space`
- `keep_embedded_space`

Setting any of the above to 1 disables that category of space trimming. Setting them to 0 restores the trimming. Examples follow.

The following code causes all leading, trailing, and embedded spaces to be retained and used for sorting.

Figure 99 Retain word spaces in index entries

FOSI fragment

```
<usetext source="index.app" userule="2" useparam="keep_all_space=1">....
```

The following setting prevents leading spaces from being trimmed so they are used for sorting.

Figure 100 Retain leading spaces in index entries

FOSI fragment

```
<usetext source="index.app" userule="2" useparam="keep_leading_space=1">....
```

The last example causes only embedded spaces to be trimmed. Leading and trailing spaces would be retained and used for sorting purposes.

Figure 101 Trim embedded spaces

FOSI fragment

```
<usetext source="index.app" userule="2" useparam="keep_all_space=1 keep_embedded_space=0">....
```

Leading punctuation and sorting (sortleadpunc)

When leading punctuation is followed by a character other than a space, the punctuation is ignored for sorting.

For example, to specify that leading punctuation should always be used for sorting, set the `sortleadpunc` useparam to a nonzero value in the `usetext` that outputs the index string variable.

Figure 102 Use leading punctuation for sorting

```
<usetext source="index.app" userule="2" useparam="sortleadpunc=1">
```

Markup in index locators (locmarkuppriority)

Locators can be highlighted to indicate the first use of the term, or a reference to a table or figure, etc. The `locmarkuppriority` useparam is used for this purpose and others. It can be set to specify the following:

- How a locator appears if an entry occurs more than once on a page and one of the locators is to be highlighted.
- How a highlighted locator is handled when it falls within a range of locators.
- Whether highlighted locators are listed in order with other locators or if they appear first.

The value of the `locmarkuppriority` is a list of the element names and/or a keyword:

- `#NONE` applies to locators with no markup specified.
- `#OTHER` applies to all other markup found around a locator, but not specified in the list of elements.

NOTE: If `#OTHER` is not included in the `locmarkuppriority` useparam, elements found around a page locator, but not mentioned explicitly in the list of element names, generate an error message.

Any markup, `#NONE`, and `#OTHER` are separated by a left angle bracket (`<`) or a comma (`,`). These separators are used as follows:

- “`<`” determines the order in which types of locators appear and which can be grouped together.
- “`,`” determines the priority of the appearance of the locator when an entry contains duplicate locators with different markup.

The following table provides some examples.

Table 15 Locmarkuppriority examples

locmarkuppriority	Notes
'bold < #NONE'	All locators in <code><bold></code> appear before locators with no markup
'bold, italic, #OTHER'	Locators in bold, italic, or any other markup can appear in any order. <code><bold></code> and <code><italic></code> locators on the same page appear <code><bold></code> . Locators in <code><italic></code> and other markup (besides <code><bold></code>) on the same page are formatted in <code><italic></code> .
'bold, italic < #NONE, #OTHER < large'	Locators in <code><bold></code> or <code><italic></code> appear before all other locators. <code><bold></code> and <code><italic></code> locators on the same page appear in <code><bold></code> . Locators with no markup or markup other than bold, italic, or large follow. Locators without markup take precedence when duplicator locators occur for an entry. Locators in <code><large></code> follow.

Locator merging with locmarkuppriority

Locators with different types of markup are not merged. For example, `locmarkuppriority='bold, #NONE'` specifies that `<bold>` locators and locators without markup that occur on the same page appear in `<bold>`. This means that `<bold>` locators and locators without markup are not merged into ranges. The tables below show some examples.

Table 16 `Locmarkuppriority='bold, #NONE'`

All locators for the entry	After merging
1, <code><bold>2</bold></code> , 3	1, <code><bold>2</bold></code> , 3
1, 2, <code><bold> 3</bold></code> , <code><bold> 4 </bold></code> , 5, 6	1-2, <code><bold>3-4</bold></code> , 5-6
1, 2, 3, <code><bold>3</bold></code> , <code><bold>4</bold></code> , 5, <code><bold>5</bold></code>	1-2, <code><bold>3-5 </bold></code>

Ranges produced with `<ixstart>` and `<ixend>` are handled differently. When a highlighted locator occurs within a forced page range, it is displayed after the range.

Table 17 `Locmarkuppriority='bold, #NONE'`

All locators for the entry with forced ranges	After merging
<code><ixstart>1</ixstart></code> , <code><bold>2</bold></code> , <code><ixend>3</ixend></code>	1-3, <code><bold>2</bold></code>

Language for sorting (language)

The default language for sorting is English. However, the `language useparam` can be used to specify a different language for sorting, which enables different group, collation, and character entity configuration files to be used. See **Supported index configuration languages** on page 172 for values that can be entered for the `language useparam`.

For example:

```
<usetext source="greek-index.app" userule="2" useparam="language=Greek">
```

NOTE: The `APTIXLANG` environment variable can also be used to specify the default sorting language. Refer to for details.

See and See Also placement (seeorder)

The default behavior for “see also” references is to follow the locator for the entry level in which they occur. The `seeorder` useparam can be coded in the `usetext` that outputs the index to change that position. `seeorder` settings are as follows:

- 0
- 1 (the default)
- 2

See and See Also with page numbers (allowseenotalone)

...

Maximum number of page references (maxlocators)

Arbortext Editor indexing has no limit to the number of page references supported for each entry. However, a maximum number of page references for an entry can be specified with the `maxlocators` useparam. For example:

Figure 103 Specify maximum number of page references for an index entry

```
<usetext source="index.app" userule="2" useparam="maxlocators=10">
```

NOTE: If the limit is exceeded, a warning message is displayed, but all page references appear.

Korean language jaeum sorting (Korean_mode)

To enable Korean language jaeum sort, set `useparam` to `Korean_mode`: For example:

Figure 104 Enabling Korean language jaeum sort

```
<usetext source="index.app" userule="2" useparam="Korean_mode=n">
```

Chinese language Pinyin sorting (Chinese_mode)

...

Index formatting for HTML output (htmlpgrffmt)

...

Renaming index pseudo-elements

DTD element names could use the default pseudo-element names used in index processing, so it is possible to rename preliminary and final index markup. To change the name of an indexing pseudo-element, `useparam` is coded with the same name as the pseudo-element followed by an equals sign (=) and the new name.

For example, all references to `<ixpt>` and `<ixentry>` are replaced with `<prelim-head1>` and `<final-head1>`, respectively, with the following `useparam`:

```
<usetext source="index.app" userule="2"
useparam="ixpt=prelim-lev1 ixentry=final-lev1"></usetext>
```

NOTE: These changes need to be reflected in the `savetext` categories that save to the index string and in the `e-i-cs` for index pseudo-elements. For example:

```
<savetext
textid="index.app" conrule="<prelim-lev1>,ch-title.txt,</prelim-lev1>...>
<e-i-c gi="ixpt">...
<e-i-c gi="final-lev1">...
```

Default index configuration files

Producing an index relies heavily on classing and sorting individual index elements. When ordering text and dividing the index into sections, the classing and sorting processes make assumptions about the alphabet used in a document.

The default character order used to prepare an index in Arbortext Editor is the English and Western European character set ISO 8859–1 (ISO Latin 1). Default processing is controlled by `Default.*` files, which are located as follows:

- `Arbortext-path\lib\ixlang` on Windows
- `Arbortext-path\lib\ixlang` on UNIX

The sort order for individual index terms is defined in a collation control file that defines all characters significant to sorting. The default file is located as follows:

- `Arbortext-path\lib\ixlang\Default.col` on Windows
- `Arbortext-path\lib\ixlang\Default.col` on UNIX

The default collation file uses ISO 8859–1 order but adds additional character entities specified in ISO 8859-2. Character entities are translated into their character equivalents. In most cases, a character entity is sorted by the 8-bit

character code used in the default character set for its document. `Default.chr` contains definitions for additional characters that do not have 8-bit equivalents. The default character entities translation file is as follows:

- `Arbortext-path\lib\ixlang\Default.chr` on Windows
- `Arbortext-path\lib\ixlang\Default.chr` on UNIX

The indexing process creates the groups that appear as section headers in the composed index. One group is set up for every character that appears as the first letter in a word. Groups are not created in the final index for characters that are not used in the first position in a word. The default listing of groups is defined as follows:

- `Arbortext-path\lib\ixlang\Default.grp` on Windows
- `Arbortext-path\lib\ixlang\Default.grp` on UNIX

Customizing index configuration files

Custom index configurations can be created for languages that are not included with Arbortext Editor, or custom index definitions can be created. The collation (`.col`), character entity definition (`.chr`), or group (`.grp`) files can be customized.

NOTE: When naming custom index configuration files, follow the default naming conventions. For example, to create index configuration files for Maltese, name the files `Maltese.col`, `Maltese.chr`, and `Maltese.grp`.

You should also add the language to a custom `ixlang.cf` file. For example, you would add the following line to your custom `ixlang.cf` file for Spanish:

```
Maltese -> Maltese.col Maltese.chr Maltese.grp #Maltese
```

To create custom index configuration files, copy the default files from the `Arbortext-path\lib\ixlang` directory and make your changes. To make custom versions of these files automatically available, use the `Arbortext-path\custom\lib\ixlang` directory to load them at startup. Create a custom file for a particular locale, and put it the appropriate locale-name subdirectory of `Arbortext-path\custom\lib` (such as in `jpn` for Japanese).

The `APTIXLANGPATH` environment variable can also be used to specify the path to an alternate `ixlang` directory for any language that is not defined in the default `ixlang.cf` file.

Creating a custom index collation file

The sort order for individual index terms is defined in a collation control file (`.col`) that defines all characters significant to sorting. The index collation file can be customized for languages that are not included with Arbortext Editor.

NOTE: Do not customize the index collation files distributed with Arbortext Editor. Refer to **Customizing index configuration files** above for more information.

`.col` files contain five sections, detailed below, which are used to define collation specifications.

Range (optional) — Defines the characters within a Unicode range that are considered a valid sort order. Character numbers can be expressed in decimal, hexadecimal, octal, or C programming language characters. If Range is omitted, 0–255 is assumed.

For example, `Greek.col` contains the following entries:

Figure 105

```
Range:
  0-127          # Normal ASCII
 128-255        # Extended ASCII
 0x380-0x400    # Greek alphabet
```

Charmap — Maps symbols to character numbers and defines the character set to be used for index sorting.

The structure of the basic command is as follows:

```
Charmap: iso8859-2 win_1250 unicode
```

The three components are:

1. name of the UNIX character set.
2. name of the Windows character set.
3. name of the Unicode character set.

Each of these character set names is also the first half of the file name of the character set source file that should be in the same directory as this collation file. For example, `iso8859-2.map` or `win_1250.map`.

Explicitly define any unique sort orders. For example, the Charmap section for the Czech collation file contains these explicit sort orders:

TIP

Add comments by using a pound sign (#) character, which causes whatever is on the rest of the line to be ignored. To include a pound sign as an actual character, enclose it in single quotation marks.

```
Charmap: unicodexa # Latin Extended A
*INCLUDE unicode.map
CH = 'C', 'H' : alpha
Ch = 'C', 'h' : alpha
ch = 'c', 'h' : alpha

Charmap: iso8859-2 win_1250 unicode
CH = "C" "H" : alpha
Ch = "C" "h" : alpha
ch = "c" "h" : alpha
```

The last part of the Charmap contains the definitions of the sort order for ISO 8859-1 characters which may be used in Czech and which are therefore defined in the collation file to make sure sorting is correct.

```
grave_a = 128 : alpha
GRAVE_A = 129 : alpha
tilde_o = 131 : alpha
TILDE_O = 136 : alpha
grave_o = 144 : alpha
GRAVE_O = 152 : alpha
```

The format of each item in this listing is:

name=characternumber1[,characternumber2] : type

The second character number position is reserved for double-letter combinations which sort together (for example, “ll” in Spanish).

Valid values for `type` are as follows:

- `alpha` — Use for any character that would be a valid group header in an index.
- `numeric` — Currently processed as `other`.
- `space` — Use when all leading spaces are ignored or when space is actually part of a leading punctuation.
- `punct` — Use for characters that are considered leading punctuation and are ignored if they are immediately followed by any other character. For example, a file named `.orange` would sort immediately after the word “orange” in the final index.
- `invis` — Use for characters that do not print. This is the default `type` for characters that do not appear in the character map.
- `range` — The first character of a range that is defined in the Range section.
- `other` — Use for characters that do not fall into one of the other categories.

Map (optional) — Maps one character to either one or two different characters.

For example, `Maltese.col` contains the following entries:

Figure 106 Map diphthongs and ligatures to characters for sorting

```
Map:
ae -> a, e
AE -> A, E
OE -> O, E
oe -> o, e
IJ -> I, J
ij -> i, j
```

Ignore (optional) — Specifies characters that have almost no effect on sorting. Characters of type `invis` are ignored for sort purposes. However, `invis` characters are used to determine sort order when two characters are identical except for the presence of the ignored character. This is why a file named `.orange` sorts immediately after the word “orange” in the final index, and is not intermingled with other “orange” entries.

For example, `Greek.col` contains the following entries:

Figure 107 Ignore

```
Ignore:
die, acute, diagr, sup2, sup3
```

Order — Use this optional section to specify the sort order for special characters.

For example, the following appears in `Maltese.col`:

```
(e, acute_e, grave_e, circ_e, uml_e, caron_e,
 E, ACUTE_E, GRAVE_E, CIRC_E, UML_E, CARON_E)
```

It specifies that the characters `e`, `é`, `è`, `ê`, `ë`, `ě`, `É`, `È`, `Ê`, `Ë`, `Ě`

`a`, `â`, `á`, `à`, `A`, `Á`, `Â`, `À` will sort to the same location, but if more than one of them appears, they will appear in this order: `a`, `à`, `â`, `á`, `A`, `À`, `Â`, `Á`.

The names used for the character entities are written out as they appear in the Insert Symbol window. Note that each ordered group should be enclosed in parentheses.

Any characters defined in Charmap section that are not type `invis`, and that are not referred to in the Map, Ignore, or Order sections may be assigned a primary weight by placing the `*REST` directive in the appropriate position in the Order section.

For example, to cause all such characters to sort between "a" and "b", put this sequence in the appropriate Order section: a *REST b Characters assigned a primary weight by the *REST directive are assigned a secondary weight equal to their character number. The secondary weight would determine the order of two *REST characters and ensure that they are recognized as two separate characters.

Creating a custom character entities index file

...

Creating a custom index group file

...

Supported index configuration languages

Arbortext Editor supports the index configuration languages shown in the table below: The full name or the abbreviation of any of these languages may be specified. The supported names and language codes also appear in the *Arbortext-path\lib\ixlang\ixlang.cf* file.

Table 18 Supported index configuration languages

Language		Language	
Name	Code	Name	Code
Brazilian	pt-BR	Icelandic	is
Bulgarian	bg	Italian	it
Chinese, Simplified	zh	Japanese Hiragana†	ja_hira
Chinese, Traditional	zh_TW	Japanese Katakana‡	ja_kata
Croatian	hr	Korean	ko
Czech	cs	Latvian	lv
Danish	da	Lithuanian	lt

†Hiragana is the Japanese cursive script

‡Katakana is a Japanese script used to spell words imported from other languages

continued . . .

Language		Language	
Name	Code	Name	Code
Dutch	nl	Maltese	mt
English, British	en-UK	Norwegian	no
English, U.S.	en	Nyorsk	no
Estonian	et	Polish	pl
Finnish	fi	Portuguese	pt
French	fr	Romanian	ro
French, Canadian	fr-CA	Russian	ru
German	de	Slovak	sl
German, Reformed	rde	Spanish	es
German, Swiss	de-CH	Swedish	sv
Greek	el	Turkish	tk
Hungarian	hu	—	

†Hiragana is the Japanese cursive script

‡Katakana is a Japanese script used to spell words imported from other languages

You can also create customized index configuration files....

NOTE: If the specified language is not recognized, Arbortext Editor uses the system default index language files.

NOTE: Pinyin options do not have an abbreviated language code. Arbortext Editor does not support Pinyin indexing for Traditional Chinese on Windows platforms. See **Chinese language Pinyin sorting (Chinese_mode)** on page 166 for more information when selecting the Pinyin index configurations.

Error messages

The following table lists error messages that occur with indexing, due to FOSI error, incorrect indexing configuration file, or authoring error.

Table 19 Indexing error messages

Error message	Recommended Action
ERROR (indexing): Number of levels specified in FOSI is invalid. Will use default (#).	FOSI: <code>index_levels</code> useparam is set to an invalid number (less than 1). Check the value of the useparam in the <code>usetext</code> that outputs the index.
ERROR (indexing): Unexpected contents encountered between records in preliminary index.	FOSI: <code>savetexts</code> that append entries to the index string variable may contain invalid text and/or pseudo-elements. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext conrule</code> conform to the preliminary index markup.
ERROR (indexing): Unexpected contents encountered between sub-fields in preliminary index record; entire record will be ignored.	FOSI: <code>savetexts</code> that append entries to the index string variable may contain invalid text and/or pseudo-elements. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext conrule</code> conform to the preliminary index markup.
ERROR (indexing): No index records found in preliminary index.	<p>FOSI: A <code>usetext</code> is coded to output the index, but no <code>savetexts</code> exist that append entries to the index string variable. Add <code>savetexts</code> to <code>e-i-cs</code> to append entries to the index.</p> <p>FOSI: All <code>savetexts</code> that append entries to the index string variable may contain invalid text and/or pseudo-elements, causing all entries to be ignored. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext conrule</code> conform to the preliminary index markup.</p> <p>Authoring: An index is output, but markup that saves an entry to the index string variable is not present in the document instance. Add appropriate markup or content to the document instance to add entries or remove the markup that outputs the index.</p>
ERROR (indexing) :Empty top level heading; entire record will be ignored.	<p>FOSI: A <code>savetext</code> that appends an entry to the index string variable may be occurring before the level 1 entry has a value. Check the location and placement of the <code>savetexts</code> in <code>e-i-cs</code> where entries are appended to the index to ensure that the entries are appended at the appropriate time.</p> <p>Authoring: The markup that contains the text of a level 1 entry is empty or is not present in the document instance. Add appropriate markup or content to the document instance.</p>

continued . . .

Error message	Recommended Action
WARNING (indexing): Empty subheading; any subsequent subheadings will be ignored.	<p>FOSI: A <code>savetext</code> that appends an entry to the index string variable may be occurring before the subsequent levels of that entry have a value. Check the location and placement of the <code>savetexts</code> in <code>e-i-cs</code> where entries are appended to the index to ensure that the entries are appended at the appropriate time.</p> <p>Authoring: The markup that contains the text of a level 2 entry or higher is empty or is not present in the document instance. Add appropriate markup or content to the document instance.</p>
ERROR (indexing): misplaced tag <code><tagname></code> ; entire record ignored.	<p>FOSI: <code>savetexts</code> that append entries to the index string variable may contain pseudo-elements that are out of order. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext</code> conrule conform to the preliminary index markup.</p>
ERROR (indexing): misplaced <code><tagname></code> ; ignored at level <code>indexlevel #</code>	<p>FOSI: <code>savetexts</code> that append entries to the index string variable may contain pseudo-elements that are out of order. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext</code> conrule conform to the preliminary index markup.</p>
ERROR (indexing): preliminary index ended unexpectedly	<p>FOSI: <code>savetexts</code> that append entries to the index string may contain invalid text and/or pseudo-elements. Check the <code>savetexts</code> to ensure that the pseudo-elements in the <code>savetext</code> conrule conform to the preliminary index markup.</p>
ERROR (indexing): Preliminary index specifies both page locator and See for the same index term. See must be used without a page locator. See also can be used with a page locator. Delete other index entries for this index term or use See also instead of See.	<p>FOSI: If "see" references should be allowed in conjunction with page locators and/or "see also" references, set the <code>allowseenotalone useparam</code> in the <code>usetext</code> that outputs the index string.</p> <p>Authoring: The document instance contains markup that appends similar entries to the index that when merged, contain "see" references along with page locators or "see also" references. Delete markup appending certain entries or change the markup appending the similar entries so that they generate all "see" references or all "see also" references in the index.</p>

continued . . .

Error message	Recommended Action
ERROR (indexing): <ixend> has no matching <ixstart>.	<p>FOSI: A <code>savetext</code> appending an entry with the <ixend> pseudo-element exists, but a <code>savetext</code> appending the same entry with the <ixstart> pseudo-element does not exist. Add the <code>savetext</code> to the appropriate e-i-c in the FOSI.</p> <p>Authoring: The markup indicating the end of a range for an entry is present in the document instance, but the markup indicating the beginning of the range is not. Add the appropriate markup to the document instance.</p>
ERROR (indexing): <ixstart> has no matching <ixend>.	<p>FOSI: A <code>savetext</code> appending an entry with the <ixstart> pseudo-element exists, but a <code>savetext</code> appending the same entry with the <ixend> pseudo-element does not exist. Add the <code>savetext</code> to the appropriate e-i-c in the FOSI.</p> <p>Authoring: The markup indicating the beginning of a range for an entry is present in the document instance, but the markup indicating the end of the range is not. Add the appropriate markup to the document instance.</p>
WARNING (indexing): character entity <code>entity name</code> (not from the default character set) cannot be translated to sort key and will be ignored for sorting. Sorting may be defined in configuration file, for example <code>filename</code> .	<p>The character entity identified in this message is not part of the default sorting character set for the current indexing language. For information on customizing your file to include this character, refer to Creating a custom character entities index file on page 172. To change your indexing language, refer to Supported index configuration languages on page 172.</p>
WARNING (indexing): markup <tagname> cannot be translated to sort key, and will be ignored for sorting.	<p>The contents of <tagname> cannot be converted to a character in the default sorting character set for the current indexing language. For information on customizing your file to include this character, refer to Creating a custom character entities index file on page 172. To change your indexing language, refer to Supported index configuration languages on page 172.</p>
WARNING (indexing): Sort key has invalid Pinyin; will be used as is.	<p>The indexing collation routine has encountered a Pinyin character combination it does not recognize</p>
WARNING (indexing): Index term has hangul mixed with other chars; will be used as is for sorting.	<p>The indexing collation routine has encountered a non-hangul character in an index term.</p>

continued . . .

Error message	Recommended Action
WARNING (indexing): Processed Index term produces key that is too long; will be used as is for sorting.	The maximum length of a sort key is 4000 characters. If no sort key appears to be this long in your document, check the final index coding.
ERROR (indexing): Locator (page reference) not found where expected; entire record will be ignored.	Pseudo-elements in <code>savetext</code> constructs may be out of order or not conform to rules for preliminary markup. Refer to preliminary index markup for additional information.
ERROR (indexing): Can't process page key found in preliminary index.	An <code><ixpgstr></code> pseudo-element must be in the <code>savetext</code> construct, it must not be empty, and it must be strictly numeric. Refer to preliminary index markup for additional information.
ERROR (indexing): Can't use page string for page key; may need explicit key.	An <code><ixpgstr></code> pseudo-element must be in the <code>savetext</code> construct, it must not be empty, and it must be strictly numeric.
WARNING (indexing): Heading has # page location references; maximum set to # with <code>maxlocators use</code> parameter.	The <code>maxlocators optional use</code> parameter sets a maximum number of page entries for an index entry. For further information on setting this parameter, refer to <i>Specifying a maximum number of page references (maxlocators)</i> . Your document may contain unintended duplicate index entries. Check your document to verify that only intended index terms are being captured.
ERROR (indexing): Invalid record in preliminary index: no <code><ixpt></code> or <code><ixstart></code> tag found	All <code>savetext conrules</code> for index entries must contain either <code><ixpt></code> or <code><ixstart></code> to indicate the beginning of an index entry. For further information, refer to <i>Preliminary index markup</i> .
WARNING (indexing): No e-i-c found in FOSI for pseudo-tag <code><tagname></code> , used in final index. Index formatting may be affected.	The <code><tagname></code> pseudo-element is not present in the FOSI. Add an element style definition for this pseudo-element with appropriate formatting.
ERROR (indexing): Could not open collation configuration mapping file: <code>ixlang.cf</code>	Arbortext Editor could not read the current <code>ixlang.cf</code> file. For information on using this file, refer to Default index configuration files on page 167 and Supported index configuration languages on page 172.
ERROR (indexing): Collation configuration file not found in path specified by <code>APTIXLANGPATH: Pathname</code> . Using default: <code>Pathname</code> .	No collation configuration file is present in the directory specified by <code>APTIXLANGPATH</code> . Arbortext Editor will use the default collation configuration file; for further information, refer to <i>Default indexing configuration files and APTIXLANGPATH — Specifying an alternate path for index configuration files</i> .

continued . . .

Error message	Recommended Action
ERROR (indexing): Could not open default group head configuration file: Pathname/filename. Index grouping may be erroneous.	For further information, refer to Default indexing configuration files.
ERROR (indexing): Could not open default character entity sort configuration file: Pathname/filename . Some character entities may be ignored for sorting."	For further information, refer to Default indexing configuration files.
ERROR (indexing): Could not open collation configuration file:Pathname/filename. Using default file Pathname/filename instead.	For further information, refer to Default indexing configuration files, Supported index configuration languages, and Creating a custom index collation file.
ERROR (indexing): Could not open default collation configuration file: filename Index sorting may be erroneous.	For further information, refer to Default indexing configuration files, Supported index configuration languages, and Creating a custom index collation file.
ERROR (indexing): Syntax error on line # in file filename. Line ignored.	Check the syntax of any customized index configuration files. Refer to Creating a custom index collation file, Creating a custom character entities index file, and Creating a custom index group file.
ERROR (indexing): Index entry sort key exceeds allowable length #.	The maximum length of a sort key is 4000 characters. If no sort key appears to be this long in your document, check the final index coding using the techniques described in Troubleshooting an index.
ERROR (indexing): Attempt to map undefined character.	Sorting has identified a character undefined in the current character entities configuration file. For information on fixing this problem, refer to Creating a custom character entities index file.
ERROR (indexing): Required first section Charmap not found.	In the current collation configuration file. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Couldn't find required Order section.	In the current collation configuration file. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Symbol name not defined in Charmap.	In the current collation configuration file. For additional information, see Creating a custom index collation file.

continued . . .

Error message	Recommended Action
ERROR (indexing): Collation charset not specified. Using default charset.	There is no entry in the Charmap: statement that Arbortext Editor can use to beginning with "ISO" and UNIX cannot determine which character set to use when sorting the index.
WARNING (indexing): Unidentified collation charset: name Using name.	The ISO entry in the Charmap: statement points to an identifiable character set and UNIX cannot determine which character set to use when sorting the index.
ERROR (indexing): Can't find assignment in Charmap entry for character	An assignment line in the Charmap (that is, grave_a = 128 :alpha) contains no = sign.
ERROR (indexing): Bad value in filename	An assignment line in the Charmap (that is, grave_a = 128 :alpha) contains an invalid character reference number.
ERROR (indexing): Can't find type field in Charmap entry	In the current collation configuration file. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Unknown Charmap type name; alpha assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type alpha. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Unknown Charmap type name; numeric assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type numeric. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Unknown Charmap type name; space assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type space. For additional information, refer to Creating a custom index collation file
ERROR (indexing): Unknown Charmap type name; punct assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type punct. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): Unknown Charmap type name; invis assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type invis. For additional information, refer to Creating a custom index collation file.

continued . . .

Error message	Recommended Action
ERROR (indexing): Unknown Charmap type name; other assumed	The collation configuration file contains a field defining type for each character added to the collation definition. In this case, Arbortext Editor is treating the unknown character as though it were type other. For additional information, refer to Creating a custom index collation file.
ERROR (indexing): File ended within primary group	An open parenthesis with no corresponding close parenthesis occurs in the Order section of the collation configuration file.
ERROR: Invalid *PINYIN directive.	The *PINYIN directive in the Order section of the Chinese collation file contains an illegal value.

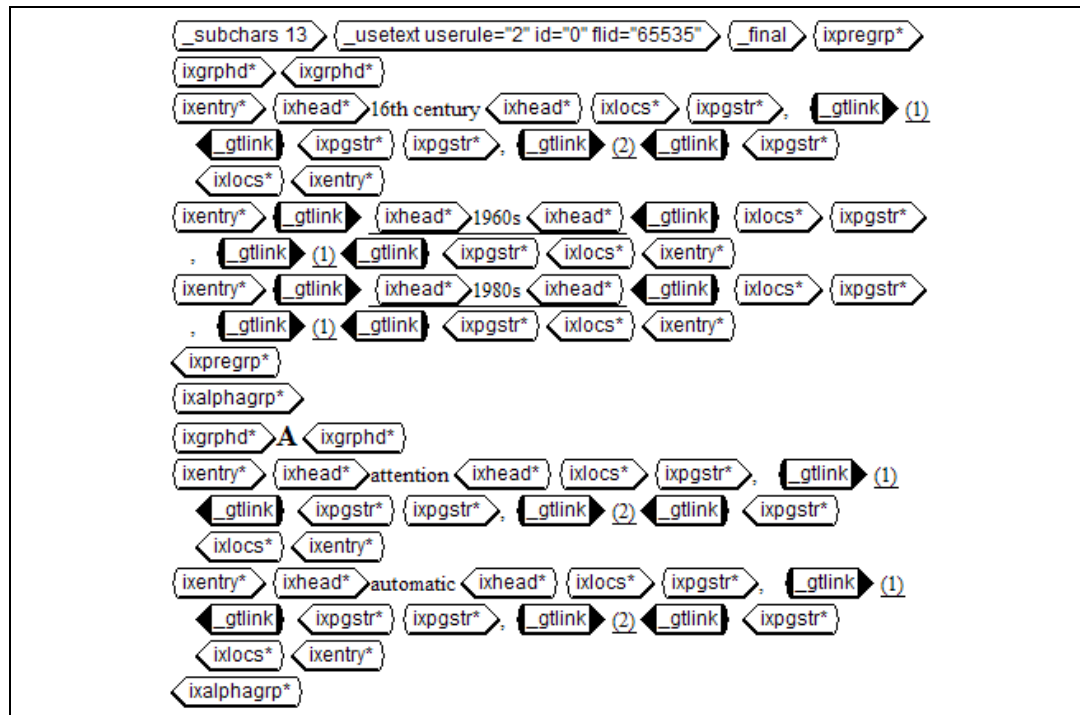
Generated sorted/merged/grouped material examples

NOTE: Examples of FOSI–sorted/merged/grouped material that make use of page numbers assume that the page number counter is incremented and saved to a `textid` in the `pageres` for each relevant page model.

In this example, the index entry element wraps text content that appears in the output stream. The first graphic in **Figure 108 Default index configuration** on page 181 shows final index in Preview. The second graphic shows a portion of the final index in the Edit window with `gentext` tags displayed.

Figure 108 Default index configuration

Index			
16th century, 1, 4	fill text, 1, 5, 9	M	recognized, 6, 9
1960s, 8	filler, 5	meaningless, 5, 9	
1980s, 8	filler text, 2, 8	Middle Ages, 6	S
	film, 8		search, 9
A	final text, 1	N	sequence, 9
attention, 5		non-readability, 2	sheet, 8
automatic, 8–9	G	nonsensical, 2, 6	sources, 6
	generate, 9		space, 1, 6
C	generator, 8	O	syllables, 2
capitalized, 5	German, 5	origin, 5–6	symbols, 5
Cicero, 6, 8	greekling, 1		systems, 9
compose, 1, 4, 6		P	
D	I	page, 2	T
designer, 1, 8	impression, 2–3	PageMaker, 8	text, 1–2, 5–9
different, 2, 5, 7		pattern, 2	typefaces, 2, 5
distribution, 2	L	popular, 8	typesetters, 1, 8
DTP, 8–9	language, 4–5	programs, 8–9	V
dummy text, 1–2,	Latin, 4–6	publication, 3, 9	versions, 7
4, 6, 8–9	layout, 2–3, 5, 8		visual, 2, 5
Dummy text, 1–2	Letraset, 8	R	
	letter, 2, 5, 7–8	random, 2	W
F	Lorem ipsum, 9	readability, 2	web, 1, 9
fill, 1, 6	Lorem Ipsum, 4–9	readable, 6	widespread, 9
		real, 1, 3–4	word, 2, 4–6



DTD fragment

```
<!ELEMENT indexterm (#PCDATA)>
```

XML fragment

```
<paragraph><indexterm>Dummy text</indexterm> is text that is used in the
publishing industry or by <indexterm>web</indexterm> <indexterm>designer
</indexterm>s to occupy the <indexterm>space</indexterm> which will later
be <indexterm>fill</indexterm>ed with "<indexterm>real</indexterm>"
content. This is required when, for example, the <indexterm>final text
</indexterm> is not yet available. <indexterm>Dummy text</indexterm> is
also known as "<indexterm>fill text</indexterm>" or "<indexterm>greeking
</indexterm>."</paragraph>
<paragraph>It is said that song <indexterm>compose</indexterm>rs of the
past used <indexterm>dummy text</indexterm>s as lyrics when writing
melodies in order to have a "ready-made" <indexterm>text</indexterm>
to sing with the melody. <indexterm>Dummy text</indexterm>s have been
in use by <indexterm>typesetters</indexterm> since the <indexterm>16th
century</indexterm>.</paragraph>
...
```

FOSI fragment

```
<stringdecl textid="index.app" status="1">
...
<e-i-c gi="indexterm">
<charlist inherit="1" charsubsetref="inline">
<savetext textid="index.app" append="1"
conrule="<ixpt>, #CONTENT, </ixpt>, <ixpgstr>, bodyfolioct.txt [B0], </ixpgstr>">
...
<e-i-c gi="ixpgstr">
<charlist inherit="1" charsubsetref="endline"></charlist>
...
<e-i-c gi="ixpt">
<charlist inherit="1" charsubsetref="startline"></charlist>
...
<e-i-c gi="ixalphagrp">
<charlist inherit="1" charsubsetref="block"></charlist>
...
<e-i-c gi="ixendpg">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\-" placemnt="before"></usetext>
...
<e-i-c gi="ixentry">
<charlist inherit="1" charsubsetref="block left">
<indent leftind="1em" firstln="0">
<presp minimum="0pt" nominal="1pt" maximum="2pt" priority="med">
</charlist>
...
<e-i-c gi="ixfinal">
<charlist inherit="1" charsubsetref="block"></charlist>
...
```

```

<e-i-c gi="ixgrphd">
<charlist inherit="1" charsubsetref="title allcaps">
<font inherit="1" size="14pt">
<leading inherit="1" lead="14pt">
<presp minimum="0.7em" nominal="0.8em" maximum="1em" condit="discard"
priority="med">
<postsp minimum="1pt" nominal="1pt" maximum="1pt" priority="high">
</charlist>
...
<e-i-c gi="ixhead" context="ixentry">
<charlist inherit="1" charsubsetref="inline"></charlist>
...
<e-i-c gi="ixlocs" context="ixentry">
<charlist inherit="1"></charlist>
...
<e-i-c gi="ixpgstr" context="* ixentry" occur="first">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\", \" placemnt="before"></usetext>
...
<e-i-c gi="ixpgstr" context="* ixentry" occur="notfirst">
<charlist inherit="1" charsubsetref="inline">
<usetext source="\", \" placemnt="before"></usetext>
...
<e-i-c gi="ixpregrp">
<charlist inherit="1"></charlist>
</e-i-c>

```

The following example is an extension of **Figure 108**. It illustrates how SYSTEM-FUNC can be used with index processing. In this case, index entries are automatically converted to lowercase letters, unless users set the <keepcase> attribute on <indexterm> to *yes*.

Figure 109 SYSTEM-FUNC for lowercase index entries

D	G	S
designers 1 dummy text 1	German 1, 6	song composers
E	L	T
European languages 1	Latin 1, 5–6 layout 1 Lorem Ipsum 1, 6–7	text 1

DTD fragment

```

<!ELEMENT indexterm (#PCDATA)>
<!ATTLIST indexterm keepcase (yes|no) "no">

```

ACL function

```
function lc(window,oid)
```

```
{return tolower(oid_content(oid))}
```

XML fragment

```
<paragraph><indexterm>Dummy text</indexterm> is <indexterm>text
</indexterm> that is used in the publishing industry or by web
<indexterm>designers</indexterm> to occupy the space which will later
be filled with "real" content. This is required when, for example, the
final <indexterm>text</indexterm> is not yet available.</paragraph>
<paragraph><indexterm>Song composers</indexterm> of the past used
<indexterm>dummy text</indexterm>s as lyrics when writing melodies in
order to have a "ready-made" <indexterm>text</indexterm> to sing with
the melody. <indexterm>Dummy text</indexterm>s have been in use by
typesetters since the 16th century.</paragraph>
<paragraph>The most well-known <indexterm>dummy text</indexterm> is the
"<indexterm keepcase="yes">Lorem Ipsum</indexterm>," which is said to
have originated in the 16th century. <indexterm keepcase="yes">Lorem Ipsum
</indexterm> is composed in a pseudo-<indexterm keepcase="yes">Latin
</indexterm> language which more or less corresponds to "proper"
<indexterm keepcase="yes">Latin</indexterm>.</paragraph>
...
```

FOSI fragment

```
<stringdecl textid="bodyfoliocr.txt" literal="">
...
<e-i-c gi="indexterm">
<charlist inherit="1" charsubsetref="inline"></charlist>
<att>
<specval attname="keepcase" attloc="indexterm" attval="no">
<fillval attname="lc" attloc="SYSTEM-FUNC" fillcat="savetext"
fillchar="conrule">
<charsubset>
<savetext textid="lc.tmp">
<savetext textid="index.app" append="1"
conrule="<ixpt>,lc.tmp,</ixpt>,
<ixpgstr>,bodyfoliocr.txt[B0],</ixpgstr>">
...
<att>
<specval attname="keepcase" attloc="indexterm" attval="yes">
<charsubset>
<savetext textid="index.app" append="1"
conrule="<ixpt>,#CONTENT,</ixpt>,
<ixpgstr>,bodyfoliocr.txt[B0],</ixpgstr>">
...
```

Figure 110 Compound page numbers in index

XML fragment

```
...
```

FOSI fragment

...

The next example illustrates the use of section numbers as locators.

Figure 111 Section numbers as locators

Index		
D	L	R
Distinct 1.5	Lorem Ipsum 2.0	Recognition 2.3
Dummy text 2.4		Repitition 1.3
E	N	T
European languages 1.4	Nonsense text 1.0	Typical text 2.2
	Nonsensical content 1.2	
G	O	
Greeking 1.1	Origin and meaning 2.1	

FOSI fragment

```

<counter initial="0" style="arabic" enumid="ATIixpagekeyct">
<counter initial="0" style="arabic" enumid="bodyfoliioct">
<counter initial="0" style="arabic" enumid="chapterct">
<counter initial="0" style="arabic" enumid="sectionct">
<stringdecl textid="ATIixpagekeyct.txt">
<stringdecl textid="bodyfoliioct.txt">
<stringdecl textid="chapterct.txt">
<stringdecl textid="sectionct.txt">
...
<rectopg>
<pageres>
<enumerat increm="1" enumid="bodyfoliioct">
<savetext textid="bodyfoliioct.txt" conrule="bodyfoliioct">
<enumerat increm="1" enumid="ATIixpagekeyct">
<savetext textid="ATIixpagekeyct.txt" conrule="ATIixpagekeyct">
</pageres>
<pagespec pgid="body.recto0">...</pagespec>
</rectopg>
<rectopg>
<pageres>
<enumerat increm="1" enumid="bodyfoliioct">
<savetext textid="bodyfoliioct.txt" conrule="bodyfoliioct">
<enumerat increm="1" enumid="ATIixpagekeyct">
<savetext textid="ATIixpagekeyct.txt" conrule="ATIixpagekeyct">
</pageres>
<pagespec pgid="body.recto1">...</pagespec>
</rectopg>
<versopg>

```

```

<pageres>
<enumerat increm="1" enumid="bodyfoliact">
<savetext textid="bodyfoliact.txt" conrule="bodyfoliact">
<enumerat increm="1" enumid="ATIixpagekeyct">
<savetext textid="ATIixpagekeyct.txt" conrule="ATIixpagekeyct">
</pageres>
<pagespec pgid="body.verso1">...</pagespec>
</versopg>
...
<e-i-c gi="index">
<charlist inherit="1">
<textbrk startpg="recto" pageid="index.page" newpgmdl="local">
<usetext source="\Index\">
<subchars charsubsetref="title one-column bold"></subchars>
</usetext>
<ruling thick="1pt" lentype="rel" relen="col" type="single">
<leading inherit="1" lead="3pt">
<postsp minimum="3pt" nominal="3pt" maximum="3pt" priority="high">
<textbrk startln="1" endln="1">
<span span="1">
</ruling>
<usetext source="index.app" userule="2">
<subchars>
<span span="3">
...
<e-i-c gi="chapter">
<charlist inherit="1" charsubsetref="start-opening-rectopg">
<reset resetlist="sectionct">
<enumerat increm="1" enumid="chapterct">
<savetext textid="chapterct.txt" conrule="chapterct">
<savetext textid="sectionct.txt" conrule="sectionct">
...
<e-i-c gi="section">
<charlist inherit="1" charsubsetref="block">
<enumerat increm="1" enumid="sectionct">
<savetext textid="sectionct.txt" conrule="sectionct">
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
...
<savetext textid="index.app" append="1"
conrule="<ixpt>,#CONTENT,</ixpt>,
<ixpgstr>,chapterct.txt,\\.\\,sectionct.txt,</ixpgstr>,
<ixpgkey>,ATIixpagekeyct.txt[BO],</ixpgkey>">
<usetext source="chapterct.txt,\\. \" placemnt="before"></usetext>
...
<e-i-c gi="title" context="section">
<charlist inherit="1" charsubsetref="title">
...
<usetext source="chapterct.txt,\\.\\,sectionct.txt,\\. \" placemnt="before">
</usetext>

```

```
<savetext textid="index.app" append="1"
conrule="<ixpt>, #CONTENT, </ixpt>,
<ixpgstr>, chapterct.txt, \.\, sectionct.txt, </ixpgstr>,
<ixpgkey>, ATiixpagekeyct.txt [B0], </ixpgkey>">
...
```

In this example, the index consists of titles in the document. Unlike a table of contents, which lists document titles in page order, in the index the titles are listed alphabetically.

Figure 112 Document titles in index

Index	
<p>A Automatic recognition of Lorem Ipsum, 6</p> <p>E European languages, 1</p> <p>G Greeking, 1</p> <p>I Incomprehensible, 5 Its function as a filler, 4</p> <p>L Lorem ipsum, 3</p> <p>N Nonsense text, 1</p>	<p>O Origin and meaning, 2 Origin and meaning of the Lorem Ipsum text, 5</p> <p>R Recognition, 2</p> <p>S Sed fermentum, 7 Suspendisse, 7</p> <p>T The usefulness of nonsensical content, 4</p> <p>V Vestibulum, 4 Vestibulum nec tortor a enim volutpat mattis habitasse, 7</p>

...

...

Figure 113 Suppressed index entry in page headers

XML fragment

```
<chapter>
<title>Planting <indexentry>vegetables</indexentry></title>
...
```

FOSI fragment

```
<rectopg>
...
<pagespec>
<header>
<usetext source="recto-header.txt [BO]"
...
<e-i-c gi="indexentry">
<charlist inherit="1">
<savetext textid="index.app" append="1" ...
...
<e-i-c gi="indexentry" context="header.ctx">
<charlist inherit="1" charsubsetref="SUPPRESS"></charlist>
...
<e-i-c gi="title" context="chapter">
<charlist inherit="1" charsubsetref="title">
...
<savetext textid="recto-header.txt"
conrule="<header.ctx>, #CONTENT, </header.ctx>">
...

```

Figure 114 Generating “Cont’d” at column breaks in an index

Figure 115 Eliminating page numbers for upper level entries

Figure 116 Sorting by a subsequent level heading